UNITED STATES AIR FORCE
RESEARCH LABORATORY

Modeling with Perceptual and Memory Constraints: An EPIC-Soar Model of a Simplified Enroute Air Traffic Control Task

James Rosbe
Ronald S. Chong
David E. Kieras

Soar Technology, Inc.
317 North First Street
Ann Arbor, MI 48103-3301

January 2001

Final Report for the Period October 1998 to January 2001

20030909 067

NOTICES

When US Government drawings, specifications or other data are used for any purpose other than a definitely related Government procurement operation, the Government thereby incurs no responsibility nor any obligation whatsoever, and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications or other data, is not to be regarded by implication or otherwise, as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

Please do not request copies of this report from the Air Force Research Laboratory. Additional copies may be purchased from:

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161

Federal Government agencies registered with the Defense Technical Information Center should direct requests for copies of this report to:

Defense Technical Information Center
8725 John J. Kingman Rd., Ste 0944
Ft. Belvoir, VA 22060-6218

DISCLAIMER

This Technical Report is published as received and has not been edited by the Air Force Research Laboratory, Human Effectiveness Directorate.
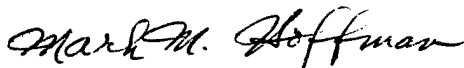
TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2002-0231

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

MARK M. HOFFMAN
Deputy Chief
Deployment and Sustainment Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 2001	Final - October 1998 - January 2001

4. TITLE AND SUBTITLE

Modeling with Perceptual and Memory Constraints: An EPIC-Soar Model of a Simplified Enroute Air Traffic Control Task

5. FUNDING NUMBERS

C: F33615-99-C-6005
PE: 63106F
PR: 2745
TA: 04
WU: 05

6. AUTHOR(S)

James Rosbe, Ronald S. Chong, David E. Kieras

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Soar Technology, Inc.
317 North First Street
Ann Arbor, MI 48103-3301

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory, Human Effectiveness Directorate
Deployment and Sustainment Division
Air Force Materiel Command
Sustainment Logistics Branch
Wright-Patterson AFB OH 45433-7604

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

AFRL-HE-WP-TR-2002-0231

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This document reports the human performance model developed by Soar Technology, Inc. as part of the initial model comparison process of the Agent-Based Modeling and Behavioral Representation (AMBR) program being conducted by the Air Force Research Laboratory. The Soar effort is a simplified enroute air traffic control (ATC) task. A human subject (or the model) performs the task of a controller by communicating with adjacent traffic controllers and aircraft to maintain a smooth flow of traffic into and out of the center airspace. "Smooth flow" is defined as performing aircraft and ATC transaction in a timely manner such that aircraft do not enter a "hold" pattern because they have not been transacted. The architecture used is a hybrid system called EPIC-Soar. EPIC (Executive Process-Interactive Control) is an architecture whose primary goal is to account for detailed human dual-task performance. Soar is a general architecture for building artificially intelligent systems and for modeling human cognition and behavior. EPIC-Soar is an integration of the perceptual and motor processor models of EPIC and Soar. It is an attempt to get the best of both worlds: the detailed predictions and explanations of sensory and motor systems from EPIC, and the broader, cognitive problem solving, planning, and learning capabilities of Soar.

14. SUBJECT TERMS			15. NUMBER OF PAGES
Human Behavior Representation Human Performance Models Joint Synthetic Battlespace Simulation			42
			16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

THIS PAGE INTENTIONALLY LEFT BLANK

PREFACE

This research was accomplished as part of the Agent-Based Modeling and Behavioral Representation (AMBR) project, managed by the Air Force Research Laboratory, Human Effectiveness Directorate, Sustainment Logistics Branch, contract number F33615-99-C-6005. The period of performance for this task was from October 1998 through January 2001.

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

1.0 INTRODUCTION

A recent National Research Council (NRC) report noted that there was a tremendous gap between the military's need for sophisticated behavioral models for training, system analysis, and command decision aiding, and the current state of the art in human performance modeling (Pew & Mavor, 1998). This report went on to state that the modeling of cognition and action of individuals and groups was possibly the most difficult task ever undertaken by humans. The report called for the establishment of long-term research programs to address this shortfall and recommended that such programs focus on modeling the realistic behavior of individuals by representing the processes (e.g., attention, memory, etc.) underlying observable behavior.

In response, the Air Force initiated a multi-year research effort with the goal of developing advanced algorithms and techniques for modeling human cognition and action, and to begin addressing the serious issues of model validation. Their approach is to fund several model developers to construct models of human behavior and to perform annual model comparisons.

The focus of the first model comparison was multiple-task behavior and subjective workload where the models would be evaluated on how well they resonated with psychological theory, fit the empirical data, and provided predictions or explanations of behavior. This document reports on the human performance model developed by Soar Technology, Inc.

Figure 1: The enroute air traffic control task environment. Our EPIC-Soar eyeball and mouse pointer are overlaid on the environment.
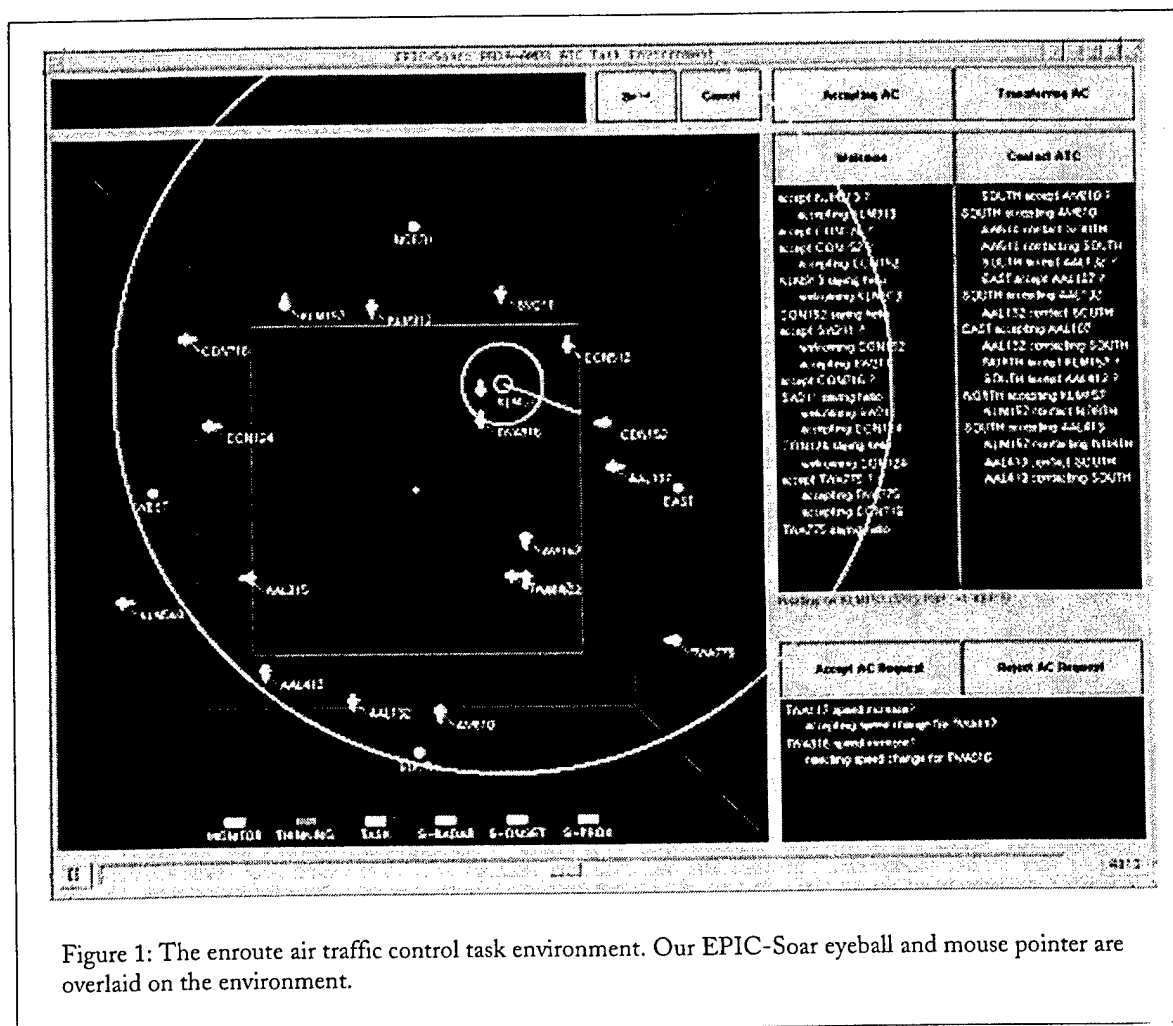
2.0 THE TASK

The task is a very simplified enroute air-traffic control (ATC) task. A human subject (or the model) performs the task of a controller by communicating with adjacent traffic controllers and aircraft to maintain a smooth flow of traffic into and out of the center airspace. "Smooth flow" is defined as performing aircraft and ATC transactions in a timely manner such that aircraft do not enter a "hold" pattern because they have not been fully transacted.

2.0.1 THE ENVIRONMENT

The task environment, shown in Figure 1, is composed of a radar display on the left, three message history lists on the right, and eight command specification buttons. The small region above the radar display is the command composition region that provides feedback as the user performs the mouse clicks needed to compose a command.

The outer square on the radar display depicts the boundaries of the center airspace. Adjacent air-traffic control centers appear to the north, south, east, and west of the center airspace. Our model implements the controller for the center airspace while the simulation environment models the adjacent controllers. One of the many simplifying assumptions of this task is that aircraft travel is constrained to be north, south, east, or west; i.e., aircraft do not travel diagonally.

2.0.2 TASK INTERACTIONS

There are three classes of transactions that are performed on aircraft. *Incoming* transactions are performed on incoming blips. The messages associated with incoming transactions appear in the leftmost message history list. An example of the transactions used for incoming aircraft is as follows:

- As an aircraft in an adjacent airspace is about to leave that airspace, the adjacent ATC will ask the center ATC to accept the aircraft, causing a message to appear in the incoming message history list, e.g., ACCEPT KLM913? as seen in Figure 1.
- The center ATC (the subject or model) must respond by accepting the aircraft before it enters the center airspace, i.e., crosses the center airspace border, otherwise the aircraft will enter a holding pattern, indicated by the aircraft turning RED. A response is composed by clicking on the ACCEPTING-AC button, followed by clicking on the incoming aircraft (KLM913), then clicking on the ATC making the request (NORTH), then clicking the SEND button. A message reflecting this response will appear in the incoming message history list, e.g., ACCEPTING KLM913.
- At some unknown time after the aircraft has entered the center airspace, the aircraft will greet the center ATC, e.g., KLM913 SAYING HELLO, creating a message in the incoming message history list.
- In response, the center ATC welcomes the aircraft by clicking on the WELCOME button, followed by clicking on the aircraft (KLM913), then clicking the SEND button. A message reflecting this response will appear in the incoming message history list, e.g., WELCOMING KLM913.

Outgoing transactions are performed on aircraft leaving the center airspace. Messages associated with outgoing transactions appear in the rightmost message history list. An example of the transactions used for outgoing aircraft is as follows:

- After an outgoing aircraft in the center airspace crosses the "transferable" border (the inner border on the radar display), the center ATC is allowed to perform a transfer command, generating a request that an adjacent ATC accept the outgoing aircraft. The request is composed by clicking on the TRANSFERRING-AC button, followed by clicking on the outgoing aircraft (AW610), then clicking on the ATC where the outgoing aircraft is headed (SOUTH), then clicking the SEND button. A message reflecting this response will appear in the outgoing message history list, e.g., SOUTH ACCEPT AW610?.
- The destination ATC will eventually send a message to the center ATC indicating that aircraft was accepted, e.g., SOUTH ACCEPTING AW610. The message will appear in the outgoing message history list.

- In response, the center ATC tells the outgoing aircraft to contact the destination ATC. The command is composed by clicking on the CONTACT-ATC button, followed by clicking on the outgoing aircraft (AW610), then clicking on the ATC where the outgoing aircraft is headed (SOUTH), then clicking the SEND button. A message reflecting this response will appear in the outgoing message history list, e.g., AW610 CONTACT SOUTH.
- The aircraft will respond, acknowledging the command, e.g., AW610 CONTACTING SOUTH.
- All of these actions *must* be performed before the aircraft crosses the outer border otherwise the aircraft will enter a hold pattern.

Aircraft in the center airspace may request to increase its speed. Messages associated with speed request transactions appear in the speed-request message history list at the bottom of the display. An example of the transactions used for a speed request is as follows:

- An aircraft will ask the center ATC for clearance to increase it speed, e.g., TWA117 SPEED INCREASE?
- The center ATC will either accept or reject the request. A request must be rejected if the requesting aircraft is directly behind another aircraft that is flying in the same direction. The request must be accepted in all other cases.
- The response is composed by clicking the ACCEPT-SPEED-REQUEST or the REJECT-SPEED-REQUEST button, followed by clicking on the requesting aircraft (TWA117), then clicking the SEND button. A message reflecting this response will appear in the speed-request message history list, e.g., ACCEPTING SPEED CHANGE FOR TWA117.

2.0.3 PERFORMANCE SCORING

Performance on the task is scored as a function of the smoothness of the flow of aircraft (as defined above) and the responsiveness of the model to requests from controllers and aircraft. The performance goal is to minimize the number of points since points are acquired as penalties. The scoring is as follows:

- Penalty for turning RED (entering a hold pattern): 50 points for each aircraft that turns red
- Penalty for staying RED: 10 points per minute
- Penalty for not answering speed request (accepting or rejecting): 2 points per minute
- Penalty for answering the speed request incorrectly: 50 points
- Penalty for not welcoming aircraft: 1 point per minute
- Penalty for duplicating a message: 10 points
- Penalty for extraneous clicks: 10 points
- Penalty for incorrect messages: 10 points.

Empirical data from human subjects were collected and used as a basis for model evaluation.

3.0 MODELING APPROACH

We have developed a very detailed, low-level performance model—at the level of eye and hand movements and human memory limitations. Our reason for modeling at this level was based on two factors: (a) the focus of the model evaluation, and (b) our observations about the task. Each of these factors will now be discussed in turn.

3.1 MODEL EVALUATION

The first factor was the dimensions along which the completed model would be evaluated. At the kickoff meeting of this contract, it was indicated that models would be evaluated by how well they were tied to psychological theory,

their level of psychological plausibility, and the behavioral efficacy of the models. These points of evaluation directed us to focus on incorporating existing psychological theory on human performance. In cases where no applicable theory existed, we have developed psychologically-inspired implementations.

3.2 Observation and Hypotheses

Like most interesting tasks that humans perform, the ATC task utilizes the following human systems: visual perception, memory, cognition, and manual motor; i.e., hands. We evaluated the role and importance of each of these systems for the task to be modeled.

It is clear that the ATC task has a strong perceptual requirement[1]. In computational terms, the eyes are responsible for finding features in the world that satisfy the preconditions of task actions. Therefore, we hypothesized that because eye scan patterns affect what can be seen and when it could be seen, perception must have a significant influence over task performance and may also be one large source of performance variability within and between subjects.

The memory system plays a key role in this task since there is generally a long delay between when an aircraft is last fixated and when an action (that relies on what is known about the aircraft) can be performed. For example, one of the important features that should be remembered is what action was last performed on an aircraft. Another is the location of an aircraft. The volatility of human memory for features such as these is readily observed in and reported by subjects. As with perception, memory effects undoubtedly influence the overall task performance and may also be a source of performance variability within and between subjects.

The knowledge in the cognitive system is inherently related to performance on the task because it provides the strategies and decision-making processes for performing the task. In addition, the cognitive system can have strategies for *coping* with the limitations of the perceptual and memory systems. Such strategies can involve frequently fixating and thinking about objects that are deemed important. Memory limitations can also be somewhat compensated for by using *memory rehearsal* (as is done for rote memorization of a phone number) to boost the strength of memories.

Unlike the previous three systems, we hypothesize that the contribution of the manual motor system to performance variability would be insignificant[2]. We came to this conclusion because once a task action sequence is triggered, its constituent steps can be performed *ballistically*; i.e., without intervening reasoning, producing roughly the same execution times regardless of the task conditions[3]. Although we hypothesize that the manual motor system's contribution would be insignificant, we did model mouse movements and clicks because these behaviors require eye-movements—eye-hand coordination is necessary to point the mouse at buttons and object—and as we have just stated, eye scan patterns are important and should be modeled.

3.3 Modeling Philosophy

These two factors—the evaluation's emphasis on psychological theory and plausibility, and our observations and hypothesis that performance on this task was strongly influenced by low-level details of the perceptual, memory, and cognitive systems—persuaded us to construct a model based on theories of perceptual, motor, and memorial capabilities and limitations, and on cognitive strategies for coping with the former. Rather than focusing on the overall performance of the model (as one would do when building a model designed specifically to satisfy the functional requirements of the task), we instead concentrated on building psychologically-plausible models of the low-level behavior, with the expectation (and hope) that plausible high-level performance representative of the human data would emerge.

---

1 Throughout this report, we make casual use of the term *perception* to refer to the sensory, perceptual, *and* ocular motor functionality of the eye.

2 We later learned that task action execution time data were collected and analyzed. No significant difference in execution times across all task conditions was found, thereby confirming our hypothesis.

3 Actually, executions times will vary slightly, depending on the distance that the eyes and the mouse must traverse to look, point, and click an object. However, this variance is *independent* of the task conditions.

There are many costs that accompany a commitment to constructing models at this level of detail. The first is that the task being modeled may include behaviors for which there is no readily available data or prominent theory in the literature. In these cases, the modeler has to do one of the following: collect new data, extrapolate from related theories, rely on observations and self-report, introspection, intuition, or best-guess.

Second, building a low-level model results in larger knowledge bases. For example, a model that is explicit about eye movement strategies will be larger than a model that assumes a generic eye scan pattern or one that does not implement an eye at all. A model that has a human-like memory will have to implement strategies for coping with memory limitations. This kind of knowledge is entirely irrelevant for a model based on a nonvolatile memory.

Third, a pragmatic cost, associated with large knowledge bases, is the time needed to debug a large knowledge base. Debugging can be made more difficult by the existence of subtle interactions between low-level components of the model and by the rare occurrence of some task situations.

Low-level modeling can (a) identify the different strategies that subjects might perform, (b) provide explanations of why one strategy may be chosen over another, and (c) make novel predictions. Ideally, one would want to thoroughly explore, evaluate, and confirm (where possible) these by-products. However, as a final cost, this is a very time-intensive process, to the extent that a "work in progress" ends up becoming a "final" model.

4.0 ARCHITECTURE

Because of the emphasis on theory and the ATC task's use of perceptual, memory, cognition, and motor systems, we felt that it was imperative to use an architecture that provided psychologically-plausible implementations of these systems. The architecture we used is a hybrid system called EPIC-Soar (Chong, 1998; Chong & Laird, 1997) that combines parts of the influential EPIC human performance architecture with the Soar cognitive architecture.

4.1 EPIC

EPIC (Executive Process-Interactive Control) (Kieras & Meyer, 1994; Meyer & Kieras, 1997a, 1997b) is an architecture whose primary goal is to account for detailed human dual-task performance. It extends the work begun with the Model Human Processor, MHP (Card, Moran, & Newell, 1983).

Like MHP, EPIC consists of a collection of processors and memories. There are three classes of processors: perceptual, cognitive, and motor. However, EPIC is distinguished from MHP in two very significant ways. First, the EPIC processors and memories are much more elaborate, each representing a synthesis of many of the most recent empirical evidence and theories describing psychological phenomena. Secondly, EPIC is a system that can be programmed and executed.

There are three perceptual processors—visual, auditory, and tactile—that receive inputs from simulated physical sensors. The visual perceptual processor, which is of particular interest for this task, represents the retinal zones (bouquet, fovea, parafovea, periphery) and the limitations of feature availability as a function of retinal zone.

The output of the perceptual processors is sent to the working memory of the cognitive processor. The cognitive processor consists of working memory, long-term memory, production memory, and a multi-match, multi-fire production system. The cognitive processor has no learning mechanism. On receiving input from the perceptual processors, it performs the reasoning necessary for the task being modeled and sends output commands to the motor processors, of which there are three: ocular, vocal, and manual.

Every EPIC model of multi-task behavior includes an executive process, encoded as productions, whose purpose is to coordinate the progress of the other processes (tasks) in the model. The executive process does not take part in accomplishing the tasks directly (such as sending motor commands in service of a task); it only mediates the access of of the competing tasks to limited resources, such as the motor processors.

4.2 SOAR

Soar is a general architecture for building artificially intelligent systems and for modeling human cognition and behavior (Rosenbloom, Laird, & Newell, 1993; Newell, 1990). Soar has been used to model central human capabilities such as learning, problem solving, planning, search, natural language, and HCI tasks.

Soar is a goal-oriented architecture, where the primitive deliberative act consists of the selection and application of operators. Soar has two representations of memory: procedural and declarative. Soar's long-term knowledge is encoded as productions, which carry out the basic functions of selecting and applying operators. Soar's current situational awareness (a combination of perceptually-generated and cognitively-created structures) are held in Soar's declarative working memory. At its core, Soar is a multi-match, multi-fire production system.

Subgoals automatically arise when the procedural knowledge is insufficient to directly select or apply an operator. In a subgoal, the basic processing recurs, so that productions in the subgoal will match to select and apply operators in service of determining which operator to select or what step to apply in the supergoal, thus resolving the subgoal.

Soar incorporates a *single* learning mechanism, *chunking*, that compiles the problem solving in the subgoals into productions. In combination with various problem solving methods, chunking has been found to be sufficient for a wide variety of learning (Lewis, *et al.*, 1990; Miller & Laird, 1996).

Soar's theories of sensory or motor processes are nascent in comparison to its cognitive processor.

4.3 EPIC-SOAR

EPIC and Soar are somewhat orthogonal architectures. EPIC has perceptual and motor processors and a non-learning cognitive processor. Soar, in contrast, has no perceptual or motor processors but is a learning cognitive architecture.

EPIC-Soar is an integration of the perceptual and motor processor models of EPIC and Soar. It is an attempt to get the best of both worlds: the detailed predictions and explanations of sensory and motor systems from EPIC (an ability Soar does not possess), and the broader, cognitive problem solving, planning, and learning capabilities of Soar (an ability EPIC does not possess). See Figure 2 for a block diagram of the architecture.

To create the hybrid architecture, we performed a "mind transplant"—EPIC's cognitive processor was replaced with Soar. EPIC and Soar remain independent programs which communicate using a socket connection. Soar accepts EPIC perceptual and motor processor messages as input to its working memory, and returns motor processor commands to EPIC as output.

The cycle of interaction and information exchange between the systems is as follows: EPIC sends perceptual and motor messages to Soar and then waits; Soar accepts the inputs, runs for one decision cycle (50ms of simulated time), returns to EPIC any motor commands that may have been generated and then waits; EPIC accepts and executes any motor commands that may have been sent. This cycle repeats.

In Pew & Mavor, (1998), the following was said about the approach used to create EPIC-Soar: "This approach of incorporating the mechanisms of other architectures and models and 'inheriting' their validation against human data promises to result in rapid progress as parallel developments by other architectures emerge." (p. 95).

The attentive reader will notice that the data being modeled for the ATC task *does not* require learning. Therefore, it should be adequate to model this task using EPIC alone. Nevertheless, we chose to build our model in Soar strictly for pragmatic reasons—better usability; maturity of the system; our greater knowledge and experience with Soar; and that large knowledge bases have been demonstrated with Soar.

5.0 CHANGES TO SOAR

The EPIC-Soar hybrid architecture provides psychologically-plausible implementations of all the required systems (perception, cognition, and motor action) with the exception of memory. The memory system of Soar (and EPIC) are not psychologically plausible. This is illustrated by the following observation. In humans, forgetting is the default condition while remembering requires a volitional and effortful process—rehearsal (as in the rote memorization of a new acquaintance's name) or memorization using some highly-reliable encoding (such as creating an association between the person's name and a unique feature of their being). Soar's memory system has exactly the opposite properties: remembering is the default condition, while forgetting requires the deliberate action of removing items from memory. To resolve this dissonance, we implemented an architectural change to Soar that results in a memory system more in accordance with the reality of human memory abilities and limitations.

We added volatility to Soar's declarative memory by incorporating two ACT-R (Anderson & Lebiere, 1998) concepts: *base-level activation* and *base-level learning*. The general idea is that when an element in working memory is created, it is assigned an initial level of activation; a base-level of activation. The activation of a newly created memory
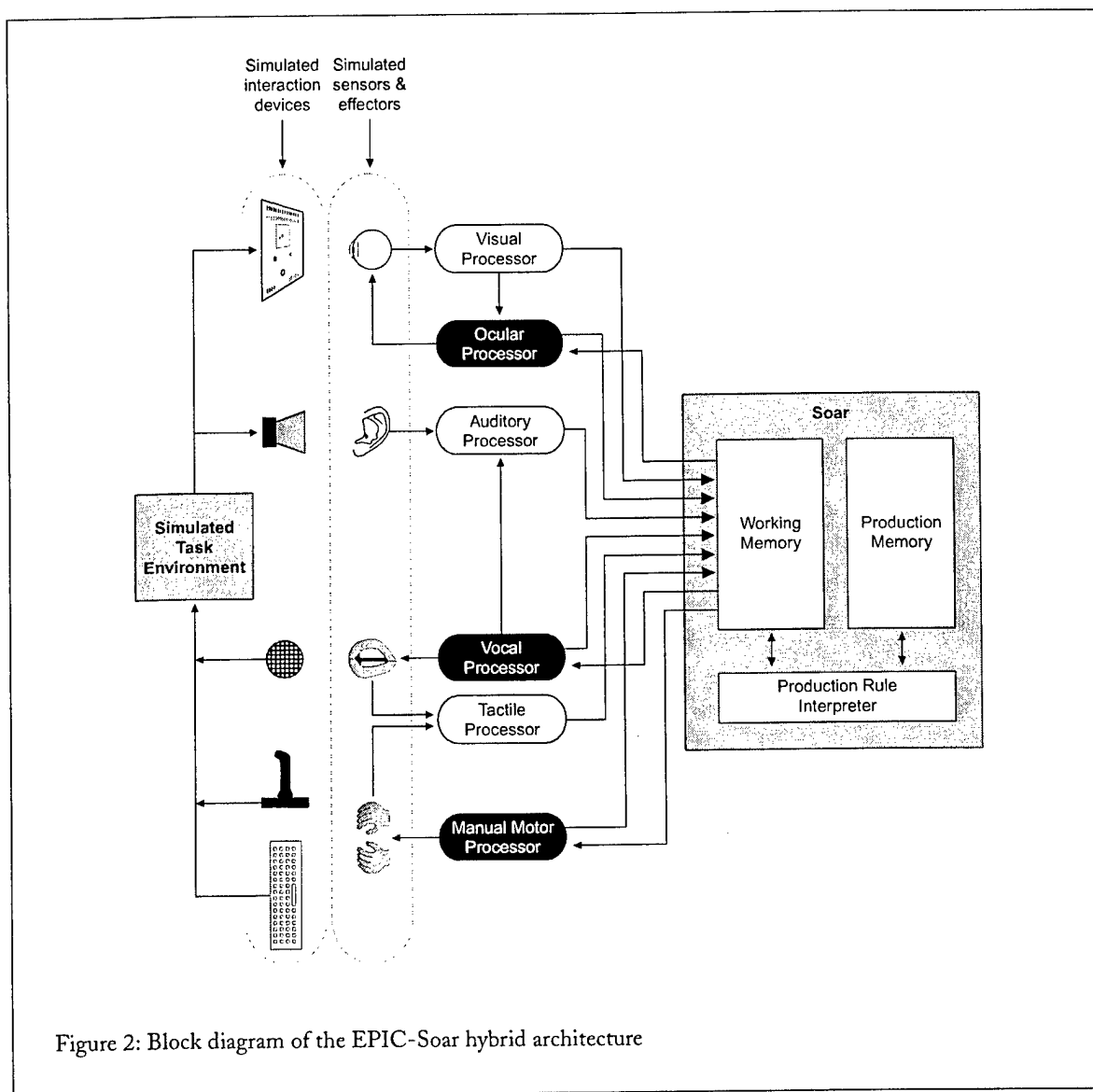
Figure 2: Block diagram of the EPIC-Soar hybrid architecture

immediately begin to decay logarithmically with time. Once the activation falls below a threshold, the element is for-gotten[4]. Forgetting is implemented by removing a decayed memory element from working memory.[5]

There are two ways to compensate for the decay process. The first is by *use* of the element in some reasoning pro-cess. Whenever an element is used, its activation receives a small boost, but again, the activation immediately begins to decay. Another way to deliberately boost activation is through *rehearsal*.

The fine line that distinguishes "use" and "rehearsal" in our model is *intent*. "Use" implies that a memory is used as part of a reasoning process that creates a useful product, such as a new memory element or the execution of a task

---

[4] The initial level of activation used in this model was 0. A BLA-DECAY-RATE parameter, set to 0.5, determines the rate at which activations decays. When an element's activation falls below a BLA-THRESHOLD parameter, set to -2.0, the element is forgotten.

[5] This removal of sub-threshold elements is a departure from the standard ACT-R theory of base-level learning. ACT-R never removes memories. Instead, forgetting effectively occurs because the decayed element does not have enough activation to make a production rule match quickly enough.

Figure 3: Dynamics of activation as a function of use/rehearsal and time (50 ms/decision)
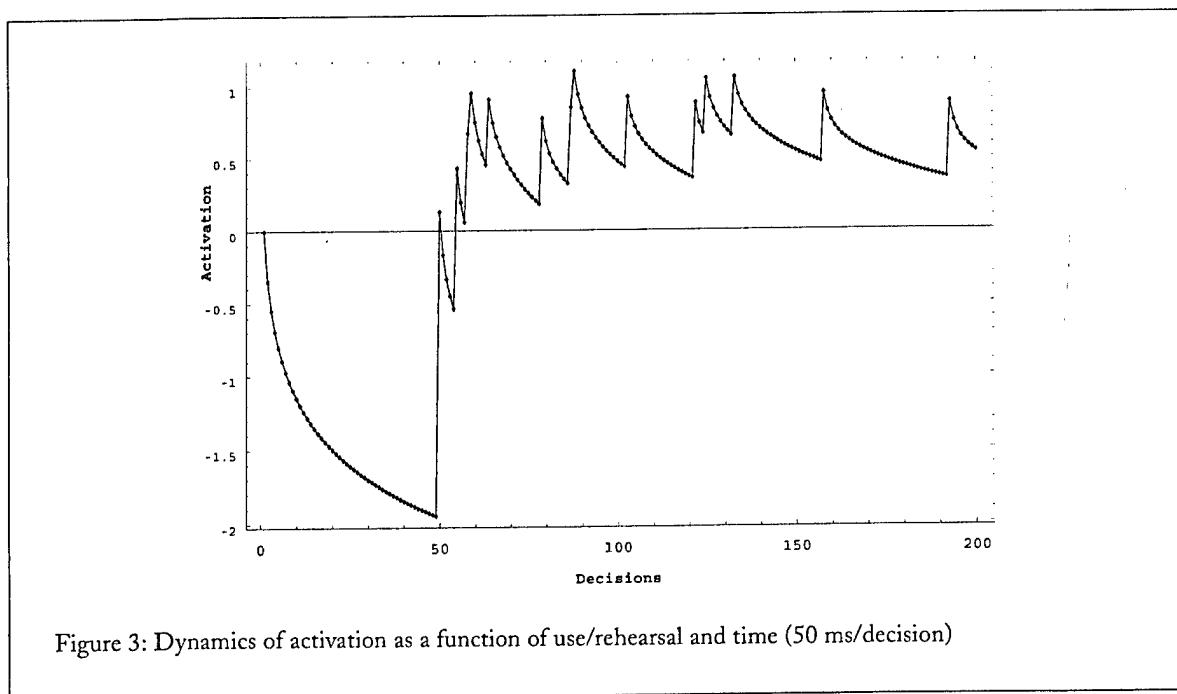
action. As such, the model's intent was to create useful products and activation is boosted merely as a kind of side-effect. In contrast, the intent in doing a rehearsal is solely to boost a memory's activation; no useful product is created. Superficially, "use" can be thought of as implicit learning while "rehearsal" is a form of explicit learning. Figure 3 illustrates that the growth and decay of activation of an element is a function of use/rehearsal (spikes in the diagram) and time.

In the base-level activation and learning theory, frequently-used (high-utility) elements will have high activations and will tend not to be forgotten. In contrast, infrequently-used (low-utility) elements will be readily forgotten.

ACT-R applies base-level activation to all chunks in its declarative memory. Our Soar implementation takes a much more reserved approach, applying activation only to task-relevant items (listed in Figure 5) that reside in a new partition of Soar's working memory that we call the *volatile memory partition*.[6] This reserved approach was used because (a) spontaneous forgetting in Soar is a new idea that might have grave ramifications if broadly applied to all of declarative memory; and (b) this approach was sufficient to attain the kinds of memory effects we felt were essential for modeling this task.

6.0 THE MODEL: AN OVERVIEW

This section will discuss three levels of representations in the model. First is the perceptual representations. Because the model interacts with a task environment, we must discuss how the perceptual information in the display enters the

---

[6] One example of the kind of items not placed in the volatile memory partition are those associated with the description and creation of goals. (In Soar terms, the proposal and elaboration of the operator.) This is done to prevent the model from forgetting what it is currently doing or performing some other kind of pathologic behavior. On closer inspection, our reserved implementation is not completely contrary to how activation is applied and used in ACT-R. For instance, ACT-R treats goals in effectively the same way as we do. When a chunk becomes a goal, its activation is "frozen" and is no longer subject to decay. In effect, the goal chunk has been moved into a logically distinct partition of memory; i.e., a non-volatile memory partition. Hence, the goal stack in both architectures represents a non-volatile memory partition.

8

model. We next discuss the declarative knowledge in the model, followed by an introductory discussion of the procedural knowledge. A more elaborate discussion of the procedural knowledge appears in Section 7.0.

6.1 PERCEPTUAL REPRESENTATIONS

The task environment notifies EPIC-Soar of the appearance/disappearance of and the basic perceivable features of blips and messages. These notifications enter the perceptual visual processor of EPIC where a mental representation of the display is created and maintained. For each type of object—blips or messages—we create additional features to those provided by the task environment.

6.1.1 BLIP FEATURES

The task environment notifies EPIC of the appearance and disappearance of blips. Additional, each time an aircraft's location on the radar display is updated, the environment notifies EPIC of the change. The basic features that are used to describe blips are: x, y, v_x, v_y, *blip-name*, and *blip-color*. Part of the mental representation of blips is based on these features:

- LOCATION: defined by the x and y position specification.
- DIRECTION: implicit in the v_x and v_y velocity specification.
- IDENT: defined by the *blip-name*.
- COLOR: defined by the *blip-color*.

While these features provide only the most basic description of the display, to properly model human perception, we need to compute additional perceptual features that people might use to perform the task. These additional features are as follows:

- PERCEPTUAL-PROXIMITY: returns a value representing how far an aircraft is from the center airspace border. This feature is dependant only on the aircraft's LOCATION.
- ABSOLUTE-PROXIMITY: returns a qualitative value (VERYFAR, FAR, NEAR, VERYNEAR) representing how far an aircraft is from the border it is flying towards. This feature depends on an aircraft's PERCEPTUAL-PROXIMITY *and* DIRECTION.
- IN-OR-OUT: indicates if an aircraft is INBOUND or OUTBOUND. This feature is dependant on the aircraft's DIRECTION and LOCATION.
- IN-TRANSFER-REGION: if an aircraft is in the transfer region (between the inner and outer borders of the center airspace), then this feature is TRUE, otherwise it is FALSE.
- IN-CENTER-SPACE: if an aircraft is in the center airspace then this feature is TRUE, otherwise it is FALSE.
- ATC: indicates the name of the ATC that an aircraft will need to interact with. If the aircraft is INBOUND, then ATC is opposite the aircraft's DIRECTION. If the aircraft is OUTBOUND, then ATC is the same as the aircraft's DIRECTION.
- CLOSEST-BLIP-IN-FRONT-IN-LINE: if there is an aircraft in front of and in-line with the aircraft, this feature holds the name of that aircraft, otherwise, the value is FALSE. This feature is used in determining the proper response to a speed request. (The computation of this feature is described in Section 7.5.)

6.1.2 MESSAGE FEATURES

The task environment also notifies EPIC of the appearance and disappearance of message history list (MHL) messages by providing the text of the message and a specification of the MHL where the message appears; INBOUND, OUTBOUND, SPEED REQUEST. As was done with blips, we compute additional perceptual features that facilitate performance on the task. These additional features are as follows:

9

- MESSAGE-TYPE: the value of this feature is a symbol that represents the semantic content of the message. Inbound message-types are: ACCEPT-REQUEST, AC-ACCEPTED, AC-SAYS-HELLO, AC-WELCOMED. Outbound message-types are: TRANSFER-REQUEST, AC-ACCEPTED, AC-TOLD-TO-CONTACT, AC-SAYS-ITS-CONTACTING-ATC. Speed request message-types are: AC-MAKING-SPEED-REQUEST, ACCEPTING-SPEED-REQUEST, REJECT-ING-SPEED-REQUEST.
- MESSAGE-OBJECT: the value of this feature is the IDENT of the aircraft referred to in the message.
- MESSAGE-ATC: if an ATC is mentioned in the message, this feature will hold its name.
- PREV-LINE: this feature points to the previous line in the message history list, if one exists. Our model searches MHLs from the bottom-up.
- TAIL-LINE: if the message is the most-recent message for an MHL, it is given a feature that indicates it is the tail-line of the list. As new messages are added, this designation is moved to the appropriate message.

6.1.3 PERCEPTUAL LIMITATIONS

The EPIC perceptual processor represents four concentric and circular retinal zones and the limitations of feature availability in those zones. The four zones are as follows:

- BOUQUET: this zone is the "hottest" region of visual perception, the very center of the retina, and has a radius θ_B of 0.5° of visual angle.
- FOVEA: this circular region has an inner radius of θ_B (0.5° of visual angle; the outer limit of the bouquet) and an outer radius θ_F that is uniformly distributed between 0.75° and 2.25° of visual angle. This radius is sampled from this range at the end of each saccade.
- PARAFOVEA: this circular region has an inner radius of θ_F and an outer radius θ_{PF} that is uniformly distributed between 5° and 13° of visual angle. This radius is sampled from this range at the end of each saccade.
- PERIPHERY: this region covers all of visual space that is *outside* the parafovea; i.e., all points with a visual angle greater than θ_{PF}.

The reason that several retinal boundaries are sampled is because there are no hard retinal zone boundaries in the human eye. Stochastically varying these radii between saccades is our attempt to "fuzzy up" the boundaries between these zones. The table in Figure 4 lists the main perceptual features of the model and indicates the retinal zones where the features are available.

Figure 4 only lists perceptual *features*, but perceptual *events* can also have a limited range of availability. Examples of such events are object onsets, color change events, and object offsets. The most relevant perceptual event in our model is the *onset* of blips and messages. Awareness of onsets is available in *all* retinal zones.

6.2 DECLARATIVE REPRESENTATIONS

Although there are many declarative structures in our model, the ones that are of greatest importance are those that reside in the volatile memory partition. This partition contains task-specific elements that we felt were subject to forgetting during performance on the task. These task-specific elements are listed in Figure 5. They fall into three broad memory classes: feature, event, and reasoned memories.

6.2.1 FEATURE MEMORIES

The first class are memories that associate an aircraft's IDENT with each of its perceptual features; i.e., each aircraft will have many *independent* memories. Having an aircraft's features distributed across several memories (rather than a single memory structure for all the features) allowed the model to capture the fragmentary nature of memory recall for aircraft features. For example, it is possible to recall the location of blip but fail to recall the direction that the aircraft was traveling.

10

```
        FEATURE                              RETINAL ZONES OF AVAILABILITY

        IDENT ..........................................................................bouquet
        DIRECTION........................................................................ <= fovea
        ABSOLUTE-PROXIMITY.......................................................... <= fovea
        CLOSEST-BLIP-IN-FRONT-IN-LINE...................................... <= fovea
        COLOR............................................................................. <= parafovea
        PERCEPTUAL-PROXIMITY ............................................... <=parafovea
        IN-OR-OUT...................................................................... <= parafovea
        IN-TRANSFER-REGION ..................................................... <= parafovea
        IN-CENTER-AIRSPACE ...................................................... <= parafovea
        ATC................................................................................. <= parafovea
        LOCATION........................................................................ all

        MESSAGE-TYPE...................................................................bouquet
        MESSAGE-OBJECT................................................................bouquet
        MESSAGE-ATC.....................................................................bouquet
        PREV-LINE ........................................................................bouquet
        TAIL-LINE ..........................................................................all
```

Figure 4: Sampling of perceptual features and associated retinal zones of availability.

```
     FEATURE MEMORIES          EVENT MEMORIES          REASONED MEMORIES
     atc                       blip-onset              anticipation
     color                     new-messages            expectation
     direction                                         blip-command
     location                                          knowledge-based-priority
     in-or-out                                         most-recently-searched-blip
     proximity
```

Figure 5: Classes of memories stored in the volatile memory partition.

6.2.2 Event Memories

As just mentioned, perceptual events, such as onsets, are perceptually available in all retinal zones. This wide availability might allow a model to have a perfect memory for onsets, thus being able to respond to each onset equally well. Humans, on the other hand, often forget about needing to attend to new onsets if the onset occurred in the midst of performing some behavior such as a task action sequence or searching for a message or an aircraft. Also, the more onsets that occur together, the greater the likelihood that some onsets will not be remembered; e.g., if four blips onset in close succession onto a busy display and to a busy operator, it is less likely that they will all be processed than if only two blips had onset.

In order to achieve this kind of behavior, we created a second class of memories for recording the occurrence of *perceptual events* such as the onset of a new aircraft or a new message in the message history lists. By having these events

11

recorded as volatile memory elements, the model will not have perfect recall of every onset since they might decay before the model is in a position to react to the onset. So although onsets are perceptually available in all retinal zones, the model cannot behave in a super-human fashion since it is may forget about onsets.

6.2.3 REASONED MEMORIES

The first two memory classes capture perceptions of the world. The third class of memories hold the products of *reasoning*. For example, after performing a task action on an aircraft, the model (and subjects, we suppose) makes a memory of having done that task for that aircraft. In the model, this is represented as a BLIP-COMMAND memory, an association between the aircraft's IDENT and the command that was just performed. Again, because the memory resides in the volatile memory partition, it is at risk of being forgotten. If a subject were to forget such a memory then realize they needed this information, they would begin an interesting sequence of behaviors (e.g., looking around the display or reading the message history lists) intended to *reconstruct* the forgotten memory[7]. Likewise, if our model forgets and later needs a reasoned memory (or any other memory for that matter), it must perform the same sort of perceptual searches in order to reconstruct the memory.

6.3 PROCEDURAL REPRESENTATIONS

Our model is composed of four classes of operators. Each class consists of one of more Soar operators. The classes are as follows:

- MONITOR-RADAR: this operator class scans the radar display. It consists of two consecutive operators: MONITOR-RADAR-FIXATE fixates on blips; MONITOR-RADAR-THINKING thinks about each fixated blip.
- SEARCH-MHL: this set of operators reads the messages in the message history lists (MHL). It consists of two independent operators: SEARCH-MHL-PROXIMITY-DRIVEN initiates a search of the lists when an aircraft has been observed to be in close proximity to a border or is RED (in a hold pattern) and the model does not remember enough about the aircraft to determine what needs to be done to the blip; SEARCH-MHL-ONSET-DRIVEN reads new messages.
- TASK-ACTIONS: this class consists of seven operators. Six of these operators are for performing the six different kinds of transactions—TRANSFERRING-AC, CONTACT-ATC, ACCEPTING-AC, WELCOME-AC, ACCEPT-SPEED-REQUEST, and REJECT-SPEED-REQUEST. The seventh operator, called DECIDE-SPEED-REQUEST, analyzes the display to determine if a speed request should be accepted or rejected.
- SEARCH-RADAR-DISPLAY: this class consist of one operator of the same name. When the location of an aircraft is unknown (either because it has been forgotten or because the aircraft has never been observed), this operator searches the radar display for the aircraft. This operator is used only in service of (i.e., as a subgoal to) one of the task-action operators.

Figure 6 shows a flowchart of the model and how these operator classes and individual operators are interconnected. These operators will be discussed much greater detail in the following section.

7.0 THE MODEL IN DETAIL: UNAIDED CONDITION

Subjects performing the ATC task were exposed to two task conditions: an unaided (or "text") condition, and an aided (or "color") condition. We developed a model for each condition.

We will first discussion each operator of the four classes presented above. Since the model of the unaided condition used all of these operators (as indicated in Figure 6), its discussion will be implicit in the discussion of each operator. The model for the aided condition will follow in Section 8.0.

---

7   An attribute of the displays in this task is that they allow almost perfect memory reconstruction. The exceptions occur when aircraft leave the display or when messages scroll off the top of their respective lists.
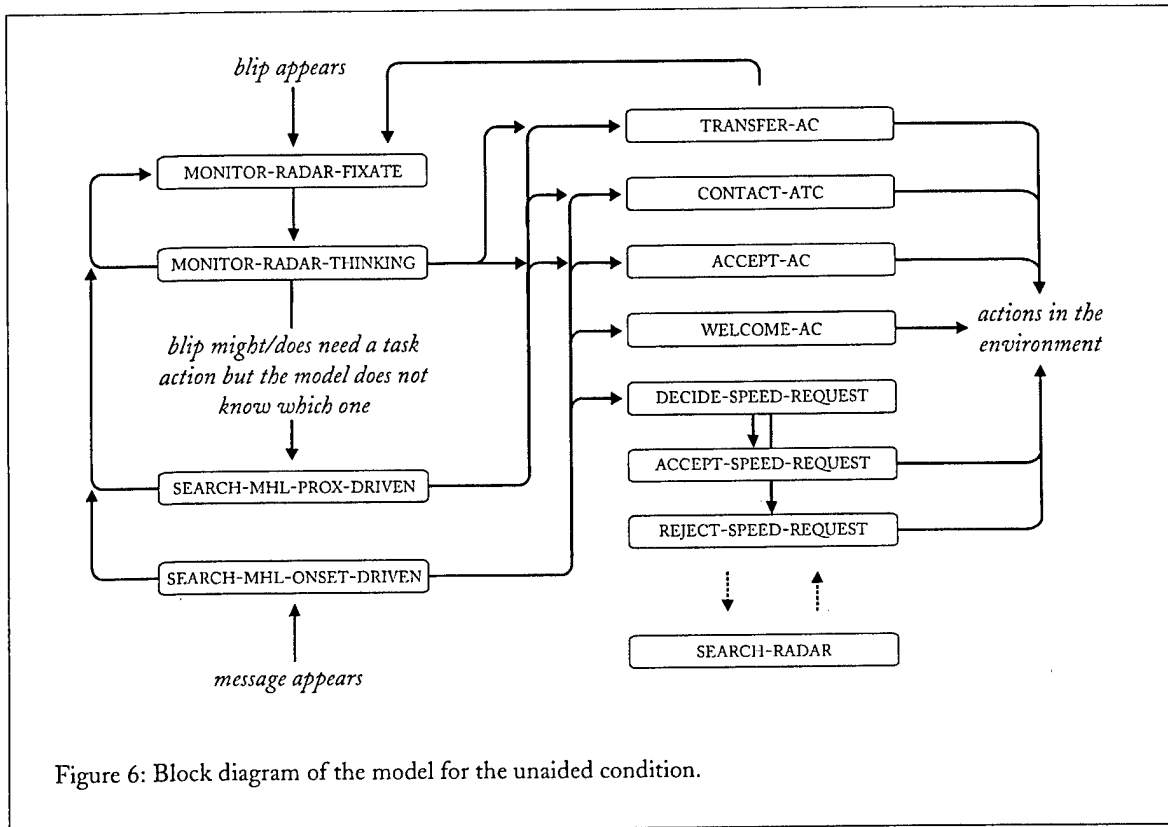
Figure 6: Block diagram of the model for the unaided condition.

Each operator is designed to capture a specific behavior that is required to perform the ATC task. Since we were building a low-level model of the task, we tried to identify and model the low-level aspects of the behavior that might be observed in subjects[8]. The discussion of each operator begins with a list of "assumptions about human behavior", followed by a detailed discussion of the operator and how those behaviors are realized.

7.1 OPERATOR: MONITOR-RADAR-FIXATE

Assumptions about human behavior

- Fixations are biased towards "important" blips.
- Possible to forget which blips are important.
- Scan patterns are not random nor systematic.
- Scan patterns are influenced by both bottom-up (reactive) and top-down processing (goal-directed).
- Scan patterns are likely to change if monitoring is interrupted.

The MONITOR-RADAR-FIXATE operator makes the model saccade to aircraft on the radar display. We conjecture that the patterns of scanning are neither random (where the next fixated object is randomly selected from what is perceivable) or systematic (where the eye follows a rigid, pre-programmed pattern that is not responsive to the dynamics of

---

[8] The data gathered for this task—performance scores and a time-tagged trace of mouse clicks—was inadequate for the level of modeling we attempted. Had eye-tracking data been collected, we might have been able to better infer low-level behaviors and strategies used by subjects. In the absence of this kind of data, we used existing psychological theory. Where none existed, we relied on inspiration, observation, reflection, and best guess.

Figure 7: Perception-based priority: regions and associate values.
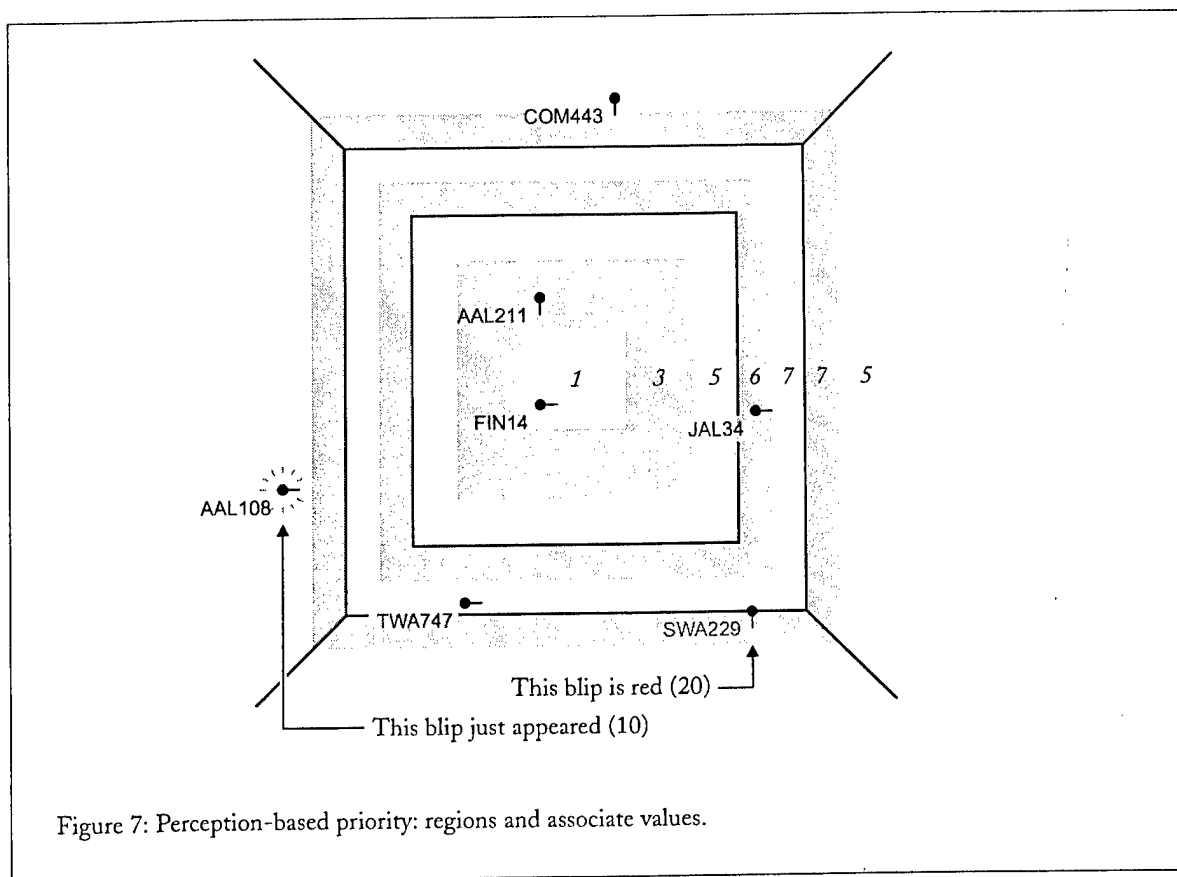
both human memory and display events). Instead, scanning seems to be reactive to the display—driven perceptually, in a bottom-up manner—while at the same time beholden to the goals and priorities of the task—driven by goals and knowledge, in a top-down manner. Therefore, our eye scanning operator is based on a combination of *perceptually-driven* and *goal-driven behavior*. This is accomplished by assigning two kinds of priorities to each blip: *perception-based priority* and *knowledge-based priority*. The model uses these two priorities of each blip to reasons which one is most important and should therefore be the next fixated blip.

The first priority is called *perception-based priority* (PBP). It is based solely on the *perceptual-proximity* of an aircraft to the border of the center airspace. An aircraft's perceptual-proximity is a feature computed by the visual perceptual processor of EPIC (see Figure 4 and associated discussion). This feature is then passed to Soar to be used in reasoning. We have defined the proximity feature to be available only in the bouquet, foveal, and parafoveal retinal regions. Therefore, an aircraft need not be fixated for its proximity feature to be made available to cognition for subsequent mapping to a perception-based priority. Aircraft located such that their proximity feature cannot be perceived (outside the parafovea, i.e., in the periphery) are assigned a default PBP of -1.

Figure 7 shows the concentric proximity regions defined in the model along with the priorities for each region. It also shows two special, non-proximity cases: (a) when an aircraft first appears, it is assigned a priority of 10, overriding the priority of five (5) based on its proximity to the outer border, and (b) when an aircraft is RED (in a hold pattern), its priority is 20, again overriding whatever priority it may be assigned due to its proximity.

The most important characteristic of this priority measure is that it is derived solely on the proximity of an aircraft to the outer border. It is therefore a rough approximation of the actual priority of the aircraft and can be inaccurate and misleading. For example, TWA747 is assigned a greater PBP (7) than JAL34 (6) even though JAL34 is closer to the eastern border and in more imminent need of attention than TWA747. The reason TWA747 has such a high priority is simply because is it so close to the southern border.
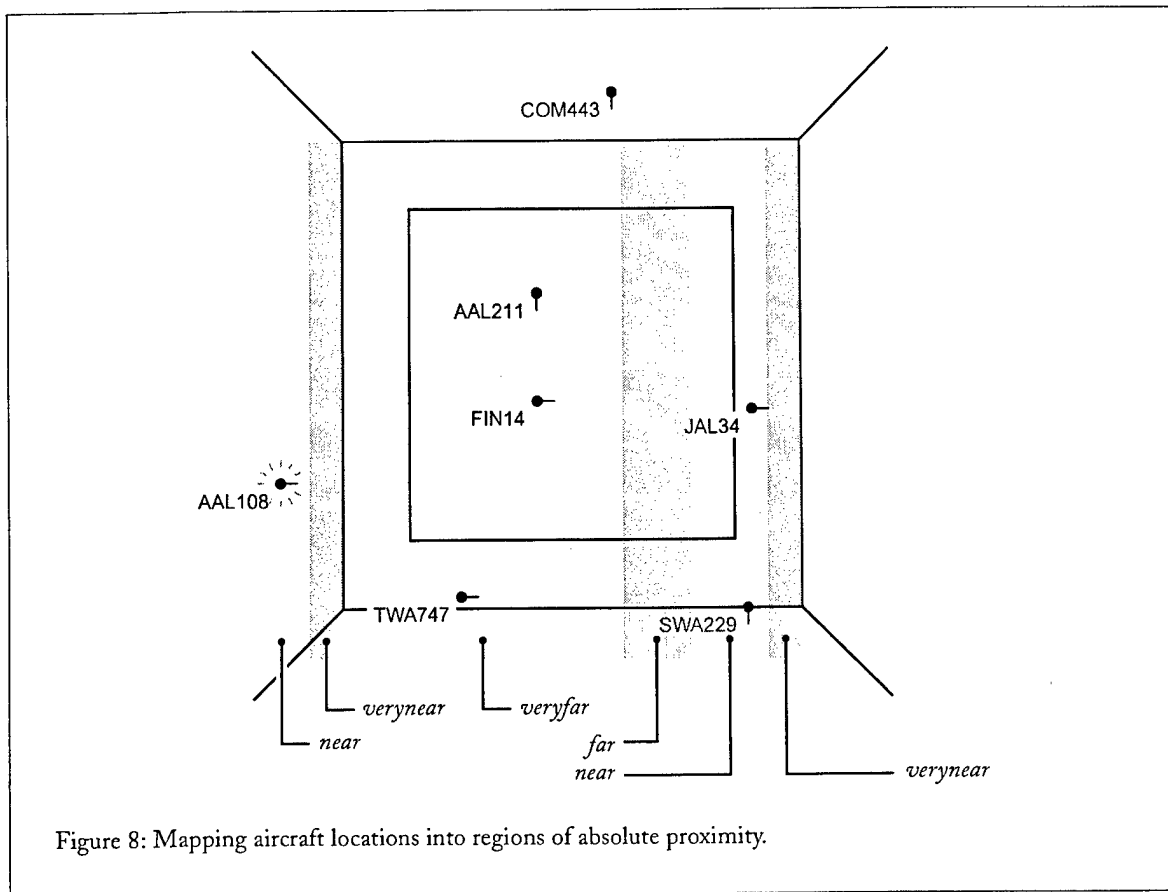
Figure 8: Mapping aircraft locations into regions of absolute proximity.

An eye scanning operator based on perception-based priority alone would produce a poor model of scanning behavior because it would be driven only by the features in the display. As an example, the model would prefer to fixate on TWA747 until JAL34 attained a higher priority; i.e., by it turning RED when it reached the border.

The second priority used in the model—knowledge-based priority (KBP)—is designed to make the scanning behavior responsive to the goals and priorities of the task. Unlike perception-based priority, which is assigned to an aircraft regardless of whether the aircraft is fixated or not, knowledge-based priority is associated only to blips that have been fixated. To compute this priority, the model considers perceptual features of the aircraft *in addition* to what is known (or not known) about the aircraft.

First, the model accesses the *absolute-proximity* feature (as seen in Figure 4, this feature is available only in the fovea and bouquet and, as discussed above, is based on both the proximity *and* direction features) that is categorized as seen in Figure 8. (The regions shown in this figure are specific to eastbound traffic. The regions for northbound, westbound, and southbound traffic can be had by counterclockwise rotation of the region map by 90°, 180°, 270°, respectively.) The model combines the absolute proximity with what it knows about an aircraft (reasoned memories) to derive its KBP. See Figure 9 for the list of rules used to assign the knowledge-based priority to inbound and outbound blips.

Consider a few examples. If the model has fixated TWA747, it discovers its absolute proximity is VERYFAR. This alone is sufficient to assign this blip a knowledge-based-priority that is very low; the application of rule 5.

If the model fixates on AAL108, it finds its absolute proximity is NEAR. Now if the model could recall that it had performed all the necessary actions for that inbound blip, then there is no need to look at AAL108 again, so the aircraft will be assigned a low KBP; the application of rule 1. If, however, the model either knew it had not performed the necessary actions for AAL108 or it could not recall what actions had been done, then AAL108 will be assigned a relatively high KBP; the application of rule 2.

```
INBOUND AIRCRAFT:

  1. If (an aircraft has been accepted) then KBP = -1
  2. If (an aircraft is not IN-CENTER-AIRSPACE, is NEAR and we expect an accept request) then KBP = 5
  3. If (an aircraft is not IN-CENTER-AIRSPACE, is VERYNEAR and we expect an accept request) then KBP = 6


OUTBOUND AIRCRAFT:

  4. If (an aircraft is FAR) then KBP = -1
  5. If (an aircraft is VERYFAR) then KBP = -1
  6. If (an aircraft is not IN-CENTER-AIRSPACE) then KBP = -1
  7. If (an aircraft was transferred and also told to contact-atc) then KBP = -1
  8. If (an aircraft was transferred but not told to contact-atc) then KBP = 7
  9. If (an aircraft has an anticipation structure and will soon be transferable) then KBP = 8



Figure 9: Rules for assigning a knowledge-based priority to a fixated aircraft.
```

As already mentioned, the model uses these two priorities to determine which blip is most important and should be fixated next. The general idea is that the blip with the highest priority will be the one next fixated. To make this determination, however, the two kinds of priorities must be taken into consideration. The model compares all candidate blips in a pairwise fashion, using the following rules:

- Rule A: If BLIP0 and BLIP1 have KBP memories, then prefer the aircraft with the higher KBP.
- Rule B: If BLIP0 has a KBP memory and BLIP1 does not have a KBP memory, then prefer BLIP1 if its PBP is greater than the KBP for BLIP0. Otherwise, prefer BLIP0.
- Rule C: If neither BLIP0 or BLIP1 have KBP memories, then prefer the aircraft with the higher PBP.

If the priority comparison yields a single "winning" blip (one with the highest priority), then it is fixated next. However, it is more likely that the comparison will yield a *set* of blips with the same highest priority. In this case, Soar randomly chooses an aircraft from this set to fixate on.

We now briefly discuss the dynamics of this operator. As mentioned in an earlier section, knowledge-based priority is a memory that resides in the volatile memory partition (see Figure 5) and is therefore subject to forgetting. As soon as a KBP is created, it will begin to decay. If the aircraft is refixated, the KBP memory will be rehearsed (provided the memory has not decayed away), boosting its activation and ensuring the memory will persist a bit longer. If the aircraft is not refixated before the memory decays, the model will forget about the knowledge-based-priority of the aircraft.

Forgetting KBPs has two effects on scanning behavior. The first occurs when the model forgets the KBP of an important blip. For example, let us assume that rule 8 in Figure 9 applies to JAL34, giving it a KBP of 7. Since the KBP is generally high, the model will tend to regularly refixate the aircraft, preventing its decay. However, KBPs can decay if the model stops monitoring the radar display and instead searches a message history list or performs a task action. Once the KBP memory has decayed, the model will not be able to refixate the previously important aircraft until its PBP is high enough to cause it to win the aircraft selection process (using Rules B or C above). Once refixated, a new KBP memory will be recreated for the aircraft.

Another effect of forgetting KBP on scanning behavior occurs for low KBP blips. In this example, let us assume that JAL34 has now been fully transacted and, by use of rule 7 in Figure 9, is assigned a KBP of -1. If the model forgets this KBP memory, the eye may be drawn away from a legitimately important blip to instead fixate on JAL34, a blip we used to know was unimportant. As above, this can only occur if its PBP is high enough to win the aircraft

selection process, as might happen if the blip is very near the border. If JAL34 is refixated, a new KBP memory will be recreated. If the model still remembers that the aircraft has been fully transacted, then a low KBP memory will be recreated, making it unlikely that the aircraft will be fixated again, at least until its KBP again decays. If the model does not remember that the aircraft has been fully transacted, then this will result in other behavior, as discussed below under the SEARCH-MHL-PROXIMITY-DRIVEN operator.

Together, the concepts of perception-based and knowledge-based priorities cause the model to produce eye scan patterns that realize our assumptions about human behavior. They provide a mechanism by which the model can balance the need to be both goal-directed yet reactive to the nature of the display. Additionally, because knowledge-based priorities are volatile, we can hypothesize that changes in scanning behavior after an interruption might be explained as an artifact of memory limitations—forgetting which blips are important.

7.2 OPERATOR: MONITOR-RADAR-THINKING

Assumptions about human behavior

- Memories are created or rehearsed
- Thinking can be preempted or aborted

After an aircraft has been fixated, its perceptual features pass through the EPIC visual perceptual processor and arrive in cognition (Soar) after the appropriate feature delays. The MONITOR-RADAR-THINKING operator creates or rehearses the *feature* and *reasoned* memories listed in Figure 5.

The process of thinking follows a pre-defined ordering, based on a *functional dependency* of the products of thinking (memories). For example, it is not possible to know the inbound or outbound status (IN-OR-OUT) of an aircraft without first knowing both its location and direction. Figure 10 shows both the functional dependency and the kinds of memories (indicated by the nodes) that are created or rehearsed during thinking. The memories on the levels of the dependency are created or rehearsed in parallel.

When the model has fixated on an aircraft and is thinking about one of its features, say IN-OR-OUT, it will create a memory for this feature if there is no pre-existing memory of this feature for the aircraft. If a memory for this feature for the aircraft already exists, then the memory is rehearsed.

In addition to creating or rehearsing memories, the model will also infer and recreate reasoned memories that have been forgotten. For instance, if the model has a memory of telling an outbound aircraft to contact an ATC, but no memory of transferring the aircraft (a prerequisite to doing the CONTACT-ATC task action), then the model infers that the transfer command must have been performed and creates a BLIP-COMMAND memory for that fact.

The top level of the dependency creates or rehearses feature memories for the basic features of the aircraft: color, direction, and location. Note that in Figure 10, the aircraft's COLOR, like its ATC feature, is memorized but is not a prerequisite to the creation of other memories. They are a prerequisite in other aspects of the model however— COLOR is used to identify and responsed to a blip in a hold pattern; ATC is used when doing most task actions.

The second level of the dependency creates or rehearses features memories for higher-level perceptual features: proximity, inbound/outbound status, and the ATC with which the aircraft will need to interact. The third and fourth levels in the dependency create or rehearse reasoned memories: anticipations, expectations, and knowledge-based priorities.

An anticipation memory is created for an outbound blip that will soon be transferable; i.e., it will soon cross the inner border. Anticipation memories help guide the eye scan patterns via the knowledge-based priority mechanism (see rule 9 in Figure 9).

An expectation memory associates an aircraft's IDENT with the *expectation of the appearance of a message relevant to the aircraft*. For example, if a new blip, AAL108, appears on the display, it receives a PBP of 10 as shown in Figure 7. Assuming this priority is sufficient for the aircraft to be fixated, the thinking operator will create many memories including the expectation that an ACCEPT-REQUEST message (e.g., ACCEPT AAL108?) will soon appear on the incoming MHL. Expectation memories are also created when task actions are performed. For example, when an outbound blip (e.g., JAL34) is transferred, an expectation is created for a message saying that the transferred blip has been accepted, e.g., EAST ACCEPTING JAL34. (The receipt of this message is very important since it triggers the CONTACT-ATC task action, completing the series of task actions needed to fully process an outbound blip.) Expectation memo-

Figure 10: Functional dependency used by MONITOR-RADAR-THINKING in the unaided condition.
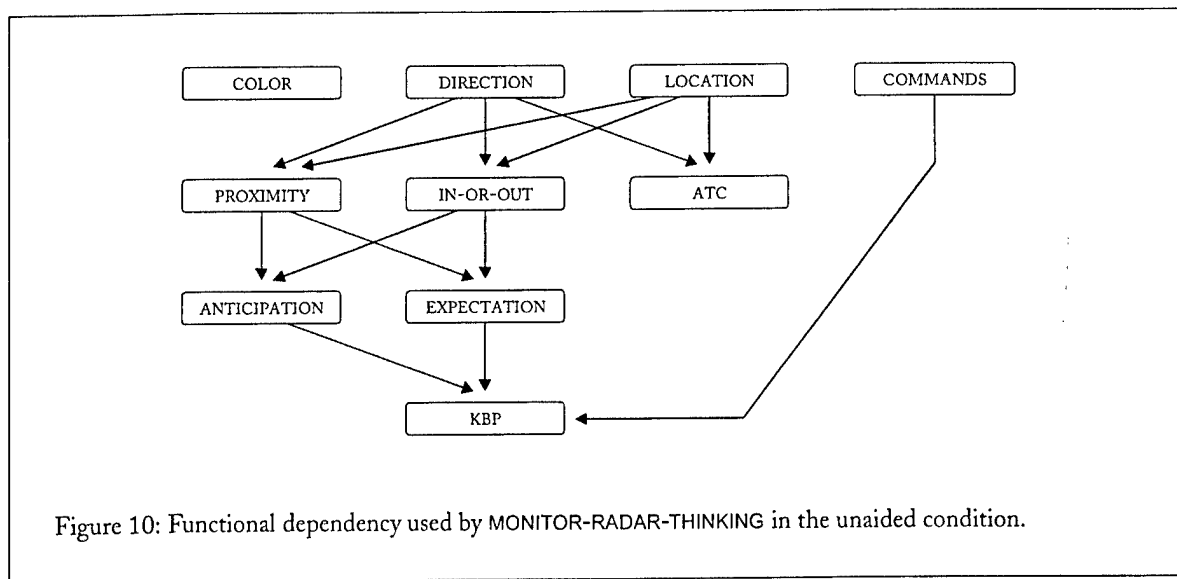
ries influence the eye scan patterns via the knowledge-based priority mechanism, as seen in the inbound rules 2 and 3 in Figure 9.

Above, we mentioned that a BLIP-COMMAND memory can be inferred from other BLIP-COMMAND memories. These memories can also be inferred from expectation memories. For example, say the model has transferred an outbound blip. After doing this task action, the model creates two memories: one for having done the transfer, and another for the expectation of an outbound MHL message from the destination ATC saying that it accepts the aircraft; a message of type AC-ACCEPTED. If at a later time, the model has no memory of transferring the aircraft, but there remains a memory for the expectation of the acceptance message, then the model can logically infer that the transfer was actually performed and can recreate a BLIP-COMMAND memory to reflect this fact. It is also through their link to BLIP-COMMANDs that expectation memories can influence the eye scan patterns via the knowledge-based priority mechanism, as seen in the outbound rules 1, 7, and 8 in Figure 9.

The final memory in the functional dependency is the knowledge-based priority (or KBP) for the fixated blip. As mentioned earlier, KBP is used to guide eye scan patterns. As indicated in Figure 10, KBP is derived from both feature and reasoned memories that appear higher in the functional dependency.

We now address the dynamics of the MONITOR-RADAR-THINKING operator. Generally, when a blip is fixated, the model steps through the levels of the dependency, creating, rehearsing, and inferring memories where possible. However, thinking can be prematurely terminated on four conditions:

- If the IN-TRANSFER-REGION feature of the fixated blip is TRUE and there is no memory of transferring the aircraft—there is no BLIP-COMMAND memory associating the aircraft's IDENT with the TRANSFERRED symbol—then the thinking operator suggests that the TRANSFERRING-AC task action be performed and immediately terminates itself.

- If the model notices a RED blip in its field of view (recall from Figure 4 that the COLOR feature is available in the bouquet, fovea), the operator will immediately terminate, relinquishing control to the MONITOR-RADAR-FIXATE operator. Because a RED blip will have a very high PBP (a value of 20; see Figure 7), it will very likely to be the next blip fixated.

- If the fixated blip's color is RED and it is inbound, the thinking operator suggests that the ACCEPTING-AC task action be performed and immediately terminates itself. The reasoning here is that an inbound blip will always ask to be accepted. The only way for an inbound blip to turn RED is if the model missed that request. Therefore the model can acknowledge the message by immediately performing the ACCEPTING-AC command.

- If the fixated blip's color is RED and the aircraft is outbound, the operator suggests that the SEARCH-MHL-PROXIMITY-DRIVEN operator (discussed below) be initiated and immediately terminates. The reasoning here is that an outbound blip requires two transactions before it is completely cleared to exit the airspace. Just seeing

18

that the aircraft is RED does not indicate what action was missed. The only way to make this determination is to read the message history list.

As a consequence of the thinking operator following the functional dependency, one possible side-effect of its ability to self-terminate is the emergence of "depth-of-processing"-like effects. Note that in the dependency, later levels of processing are dependent on (or *use*) memories that have been created in earlier levels. As indicated earlier, using an activated memory boosts its activation. Therefore, if thinking is prematurely terminated, the memories in earlier levels of processing will not be used as frequently (and hence, not have their activation boosted) as would be the case had thinking been able to proceed without interruption. The cost of premature termination is two-fold: (a) early memories will not be used as frequently and are at risk of being forgotten, and (b) the memories at the later levels of processing will not be created.

When the operator ends, the model generally executes MONITOR-RADAR-FIXATE to perform another fixation. There are two other exceptions, as shown in Figure 6. The first pertains to those instances when thinking is prematurely terminated and control is passed to one of the task action operators. The second exception arises when thinking is prematurely terminated because the aircraft *might* need or *definitely* needs a task action, but the model does not know if an action is really needed or if so, which action is needed. For this exception, the model turns to the SEARCH-MHL-PROXIMITY-DRIVEN operator to read the messages in the MHL to determine what should be done, *if* anything at all. This operator is discussed next.

7.3 OPERATOR: SEARCH-MHL-PROXIMITY-DRIVEN

Assumptions about human behavior

- Multiple search strategies
- Opportunities for memory reconstruction or rehearsal
- Non-exhaustive search
- Errors: resending messages

The first two operators discussed thus far interact exclusively with the radar display. We now discuss the first of two operators that interact with the message history lists (MHL). The purpose of the first operator, SEARCH-MHL-PROXIMITY-DRIVEN operator, is to search an MHL, beginning at the bottom, looking for a target message that matches a pattern given as input to the operator. (Note that SEARCH-MHL-PROXIMITY-DRIVEN only applies to the inbound and outbound lists, not the speed-request list.)

As we see in Figure 6, the only way to get to this operator is from MONITOR-RADAR-THINKING when the fixated aircraft *might* need or *definitely* needs a task action, but the model does not know if an action is really needed or if so, which action is needed. A fixated blip *might* need an action when the aircraft's proximity is VERYNEAR but the model does not remember if the aircraft had been sufficiently transacted. A fixated blip *definitely* needs an action when the aircraft is RED and the model does not know what action is needed to rectify this condition[9].

To summarize, these situations occur because the model has either forgotten what action/s it has performed on the aircraft or because the model has never transacted on the aircraft and does not realize it. In this knowledge-poor situation, the model needs to acquire or re-acquire more knowledge to be able to continue. This knowledge acquisition is accomplished in the same manner as human subjects; by searching the message history lists.

When the MONITOR-RADAR-THINKING operator relinquishes control to the SEARCH-MHL-PROXIMITY-DRIVEN operator, it passes a target message pattern to be found. The message pattern specifies the IDENT of the fixated blip that might or definitely needs a task action. Additionally, to know which message history list to search, the message pattern also specifies the IN-OR-OUT status (inbound or outbound) of the aircraft. The model saccades to the last message

---

[9] Note that this condition applies only to outbound aircraft. When inbound aircraft are RED, the model will immediately perform an ACCEPTING-AC command, as described earlier. This kind of certainty is not applicable to outbound aircraft since two transactions are needed to fully transact an outbound aircraft and the omission of either one will cause an aircraft to enter a hold pattern.

of the appropriate list and searches up through the list for the *most recent message* that refers to the fixated blip. There are four possible outcomes of the search:

- If the aircraft is not RED and the search fails, then the model's behavior depends on whether the aircraft is inbound or outbound:
  - If the aircraft is inbound, then the model assumes that a request to accept the aircraft has not yet arrived and returns to MONITOR-RADAR-FIXATE.
  - If the aircraft is outbound and is in the TRANSFER-REGION, then the model immediately performs a TRANSFERRING-AC action since the absence of any message for the aircraft indicates that the model has not yet performed any actions.
- If the aircraft is not RED and the search succeeds, then the model's behavior depends on whether the aircraft is inbound or outbound:
  - If the aircraft is inbound and the message is of type ACCEPT-REQUEST (e.g., ACCEPT COM443? where an adjacent ATC requests that we accept COM443 into the center airspace), then the model immediately performs an ACCEPTING-AC task action. If the message is of type AC-ACCEPTED (e.g., ACCEPTING COM443; this message is generated when the model has done the ACCEPTING-AC task action), then the model knows that nothing needs to be done for the aircraft and control is returned to MONITOR-RADAR-FIXATE.
  - If the aircraft is outbound, then the model determines from the semantics of the message what action is required. There are two possible messages type that would cause the search to succeed: (a) an AC-ACCEPTED type (e.g., SOUTH ACCEPTS SWA229) or (b) a TRANSFER-REQUEST type (e.g., SOUTH ACCEPT SWA229?). In the first case, the message indicates that the SOUTH ATC has accepted the aircraft we have transferred. The the model now knows that it can now perform the CONTACT-ATC task action. In the second case, the message indicates that the model has performed the TRANSFERRING-AC but that the destination ATC has not yet accepted the outbound blip. Since CONTACT-ATC cannot be performed, the model returns to MONITOR-RADAR-FIXATE. Note that both outbound messages explicitly or implicitly indicate one thing: that the TRANSFERRING-AC command has been performed. The model uses this fact to rehearse (or recreate) the BLIP-COMMAND memory that associates the aircraft's IDENT with the TRANS-FERRED symbol. Additionally, the second message, while indicating that the aircraft was transferred also indicates that the SOUTH ATC has not yet responded. The model uses this fact to rehearse (or recreate) an EXPECTATION memory that associates the aircraft's IDENT with the expectation for an AC-ACCEPTED message.
- If the aircraft is RED and the search fails, then the model assumes that the aircraft needs to be transferred and performs the TRANSFERRING-AC task action. (Again, proximity-driven searches for RED blips are only performed for outbound aircraft since a RED inbound aircraft indicates that an ACCEPTING-AC task action needs to be performed).
- If the aircraft is RED and the search succeeds, then behavior is exactly as the case where an outbound aircraft is not red and the search succeeds.

When searching the message history lists, our model scans across the length of a message, saccading to each word in the message, giving the appearance of reading. This detail of fixating each word may appear to be gratuitous. Nevertheless, modeling at this level of detail is justified using the same rationale we used to justify building a model using an eye with perceptual limitations: in this task, performance is highly dependant on where the eye is looking, what can be seen, and when it can be seen. Scanning across messages is no more gratuitous than scanning the radar display from blip to blip.

Modeling at this level does have its costs. One unanticipated complication was dealing with message lists that scrolled while a message was being read. In general, the additional knowledge needed to cope with this rare occurrence is straightforward. For example, all that is needed is a signal indicating that the line being read has scrolled away. With this signal, then the model needs some knowledge to find the original line and reread it.

We created a scroll signal by adding a perceptual feature called LINE-NUMBER to each message. We added knowledge to: (a) note the LINE-NUMBER of the desired message; the one the model was about to read, (b) notice when the line number of the fixated message line did not match the line number of the desired message (this occurs when the list has scrolled), and (c) fixate on the beginning of the desired message and reread the message.

Although these are seemingly simple additions, debugging the system can be difficult precisely because the scroll occurrences are rare and because of subtle interactions in the model. In our final model, there remains a very rare, fatal, and elusive bug that occurs during *some* scroll recovery attempts but not on others. On these occasions, we simply restart the simulation.

In developing our model, we implemented two strategies for doing proximity-driven searches, motivated by behaviors reported by subjects. The first strategy reads each message fully before proceeding up the list to the next message. One advantage of this strategy is that the model (and maybe subjects) can rehearse (or recreate) BLIP-COMMAND or EXPECTATION memories after reading each message (as discussed above). However, this strategy is quite slow to physically perform since each word in the message must be read. Although some memories can be created, rehearsed, or inferred with this strategy, other memories (specifically those created and rehearsed by the lower-priority MONITOR-RADAR-THINKING operator) cannot be rehearsed and are at risk of being forgotten.

The alternative search strategy—the one used in our final model—utilizes an affordance in the way messages are displayed in the task environment. The text of a message is presented in two colors: the IDENT specification is highlighted, appearing in white, while the rest of the words appear in yellow. When searching the MHL, this strategy fixates on the IDENT in the last message in the list, determines if it matches the target IDENT. If it does not, then it fixates on the IDENT in the previous message. This process repeats. If the target IDENT is found, the model then reads the entire message, word for word, to acquire the semantics of the message. The advantage of this strategy is that it takes much less time to perform. However, it foregoes the opportunity to create, rehearse, or infer memories, except for the memories due to the one message that is read entirely when the target is found.

To this point, the discussion has made no mention of the *extent* of the search. People rarely, if ever, exhaustively search the MHL. Likewise our model does not exhaustively search the lists. It is not clear what criteria people use to determine how deep to search the lists. We can speculate that it might be influenced by some combination of (a) task load, (b) remembering seeing a message on a previous search of the list, and (c) having a sense of the chronological relationship of blip locations, task actions, and message appearances. However, we took a simple approach by hard-coding a depth limit of seven (7) messages. If a target message had not been found after reading the most recent seven (7) messages, then the search would terminate with a failure.

One unanticipated side-effect of a non-exhaustive search is errors in performance. Task action duplication is considered an error in this ATC simulation. Our model can produce this kind of error. Consider the following scenario. The model notices that an outbound aircraft (say SWA229) has just entered the transfer-region and performs the TRANSFERRING-AC task action. The model then gets preoccupied performing other task actions or scanning the display. The model later notices a red blip (SWA229), fixates it, then realizes it does not remember what action needs to be performed. The model proceeds with a proximity-driven search, but the search fails—no message is found within the first seven messages—because many other messages for other outbound blips had been added to the list. Since the search failed and the aircraft is outbound, the model immediately performs a TRANSFERRING-AC action, *again*.

We have been unable to compare the frequency of these errors in subject and humans. Nevertheless, we see here an example of how unanticipated (but welcomed) behavior can emerge from combination of the dynamics of the task, modeling with perceptual and memory limitations, as well as capturing different kinds of performance strategies.

7.4 OPERATOR: SEARCH-MHL-ONSET-DRIVEN

Assumptions about human behavior

- Forgetting that a message onset has occurred
- Deliberately not reading messages
- Opportunities for memory reconstruction or rehearsal

There are only two situations where the model should ever need to look at the message history lists. The first is when it observes an aircraft that might or definitely needs attention. In this situation, the model does the SEARCH-MHL-PROXIMITY-DRIVEN operator as just described. The second situation occurs whenever a new message *appears* (or onsets) on a message history list. In response, the model will executed the SEARCH-MHL-ONSET-DRIVEN operator.

21

As we see in Figure 6, this operator is only activated when a new message appears on a message history list. When activated, the operator fixates the beginning of the new message, then scans across the length of the message fixating each word, as described in the proximity-driven search operator.

After the model has "read" the message, it determines if the message *prompts* an action. If it does, the model will immediately perform that action. For example, if an outbound message is of type AC-ACCEPTED (e.g., SOUTH ACCEPTS SWA229), then the model knows that it should immediately perform a CONTACT-ATC task action. If the message does not prompt an action, then the model rehearses (or recreates) BLIP-COMMAND and/or EXPECTATION memories for the aircraft whose IDENT is referred to in the message, then relinquishes control to the MONITOR-RADAR-FIXATE operator.

One of our assumptions about human behavior is that subjects can forget about message onsets. For example, if a subject is performing a task action and a message appears, the subject may make a mental note of the new message, but by the time the task action is complete, there is no guarantee that the subject will remember that the onset occurred and then read the message. Additionally, while doing the task action, several other new messages may have appeared. In this situation, the likelihood of the subject remembering and acting on all the onsets is very small.

Our model captures this behavior through the creation of a NEW-ONSET *event* memory (see Figure 5) for each new message that appears. These memories reside in the volatile memory partition. Therefore, the SEARCH-MHL-ONSET-DRIVEN operator is activated, not by the existence of a new message, but rather by the existence of a NEW-ONSET memory. It is by this indirection that the model can forget about onsets.

In an attempt to improve the model's performance with respect to the empirical performance data, we invoked a speculative assumption: that subjects might deliberately ignore reading some message onsets. Within this task there is good reason to consider such a performance strategy. Specifically, half the messages in the message history lists fall into the class of message we call "reminder" messages. These messages appear whenever a task action has been performed. For example, as soon as the SEND button is pressed when doing the TRANSFERRING-AC task action, a message of type TRANSFER-REQUEST (e.g., SOUTH ACCEPT SWA229?) will appear, merely indicating that the TRANSFERRING-AC task action was performed. Although these reminder messages are vitally important—memories can be created, rehearsed, and inferred based on these messages—they probably are not so important that the model should look at *every* reminder message as soon as it appears; i.e., right after the SEND button has been pressed. By ignoring reminder messages, the model can spend more time doing other tasks and possibly improve its performance.

In our initial implementation of this idea, we had the model ignore *all* reminder messages. As soon as a task action was completed, the model would ignore the next new message for the associated MHL to appear. To our surprise, this strategy appeared to yield even poorer performance. We speculated that the benefit of not having to read the reminder messages might have been overshadowed by the loss of opportunities to create, rehearse, or infer BLIP-COMMAND and EXPECTATION memories.

Our final solution was to allow the model to read the reminder messages for the inbound and outbound message history list but not for the speed-request message history list. We felt that this was an appropriate solution for a number of reasons: (a) speed request messages are the longest messages and much time could be saved if these reminder message were not read, and (b) there is no value in rehearsing speed request reminder messages since no other actions are dependant on them. This revised strategy did appear to improve performance, but we would need to do further analysis to better assess the significance of this change in contrast to the many other tweaks applied to the model as the project neared the end.

7.5 OPERATOR: TASK-ACTIONS

Assumptions about human behavior

- Eye-hand coordination
- Create a memory for having done a command
- Create an expectation of a new message for the aircraft

There are seven task actions operators. These operators utilize EPIC's manual motor processor to move a simulated mouse and to click on objects and buttons to compose the commands that are sent to aircraft and other ATCs. Before a mouse movement can be performed, the eye must either be at the destination location, or be on its way to the desti-

nation location[10]. Therefore, the time to perform mouse movements is not just the time to move the mouse, but also the time to initiate the movement of the eye.

The first four operators provide the four most commonly performed transactions: TRANSFERRING-AC, CONTACT-ATC, ACCEPTING-AC, WELCOME-AC. After performing each action, the model creates both a BLIP-COMMAND memory, associating an aircraft's IDENT with the action just performed. In addition, when the TRANSFERRING-AC and ACCEPT-ING-AC commands are given, the model creates an EXPECTATION memory for the messages that trigger the companion task actions; CONTACT-ATC and WELCOME-AC, respectively.

The least commonly performed transaction is the speed-request transaction. A speed request is a two-choice task. We have represented this as three operators: one operator to decide what to do—DECIDE-SPEED-REQUEST—and two task action operators that implement the result of the decision—ACCEPT-SPEED-REQUEST or REJECT-SPEED-REQUEST. The most interesting of these three is the first, the operator that decides what action to perform.

To determine if a speed request can be accepted or rejected, the model must answer the following questions: (a) is there an aircraft *in front* of the requesting blip, (b) if so, is the aircraft *inline* with the requesting blip, and (c) if so, is the aircraft traveling in the same *direction* as the requesting blip. If all three are true, then the speed request must be rejected. Otherwise, the request can be accepted.

In our model, we created a new perceptual feature in EPIC called CLOSEST-BLIP-IN-FRONT-IN-LINE that answers the first two questions. The value of this feature will either be FALSE or be the IDENT of the closest aircraft that is both in front and inline with the requesting blip. The behavior of the model is as follows:

- It fixates the requesting aircraft to collect the value of its CLOSEST-BLIP-IN-FRONT-IN-LINE feature.
- If the value is FALSE, the model performs the ACCEPT-SPEED-REQUEST task action.
- Otherwise, the model fixates the aircraft indicated by the CLOSEST-BLIP-IN-FRONT-IN-LINE feature to retrieve its DIRECTION feature.
- If the direction is the same as the requesting blip, then the model performs the REJECT-SPEED-REQUEST task action (presumably because the requesting blip might collide with the blip that is in front and inline).
- Otherwise, the model performs the ACCEPT-SPEED-REQUEST task action.

As mentioned, the CLOSEST-BLIP-IN-FRONT-IN-LINE feature is a perceptual feature produced by EPIC. It is computed by finding the closest aircraft to the requesting aircraft that lies within a bounding box that is twice θ_B wide ($1.0°$ of visual angle; the diameter of the bouquet) and θ_{PF} long (the outer radius of the parafovea) and oriented with the long side aligned to the travel direction of the requesting aircraft. This is depicted in Figure 11 using two examples.

In the first example of Figure 11, FIN14 has made a speed request and the model has fixated on the aircraft. A bounding box is drawn with its long side aligned parallel to the direction of travel of the requesting aircraft. In this example, we see that because JAL34 is the only aircraft that lies within the box, the value of CLOSEST-BLIP-IN-FRONT-IN-LINE for FIN14 is JAL34. Since CLOSEST-BLIP-IN-FRONT-IN-LINE for FIN14 is not FALSE, the model will fixate on JAL34 to ascertain whether or not it is traveling in the same direction as FIN14. It will find that the aircraft are traveling in the same direction and will reject the speed request.

This example shows that the "line" we use to determine "inlineness" has some thickness. If a zero-thickness line were used instead, the model would be a better judge of inlineness than humans. Therefore, we use a box to crudely approximate the *existence* (not the *magnitude*) of a range of inlineness that subjects might demonstrate in determining if two objects are inline with one another. (A better approximation might be a triangle, with the point of the acute angle at the center of the bouquet.)

The second example depicts a case where the CLOSEST-BLIP-IN-FRONT-IN-LINE feature for AAL211 is FIN14. However, when the model looks at FIN14, it will discover that the aircraft are flying in different directions. The model will then accept the speed request for AAL211. The parafovea has been reduced intentionally in this example to (a) reinforce that fact that parafovea size varies and (b) show that the length of the bounding box varies with the size of the parafovea.

---

10 This constraint is not enforced in the current version of EPIC; i.e., it is possible to move the mouse to an object without looking at the object. To implement eye-hand coordination, we deliberately program the eye to move to the object before the mouse movement is made.
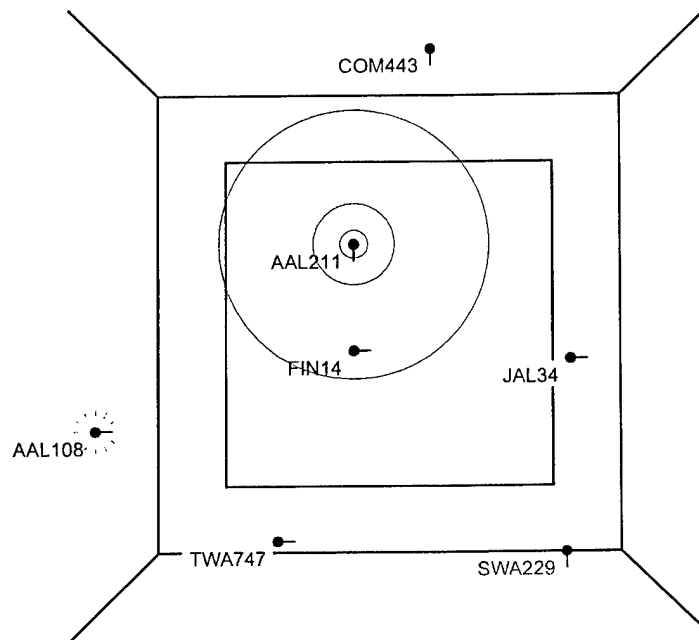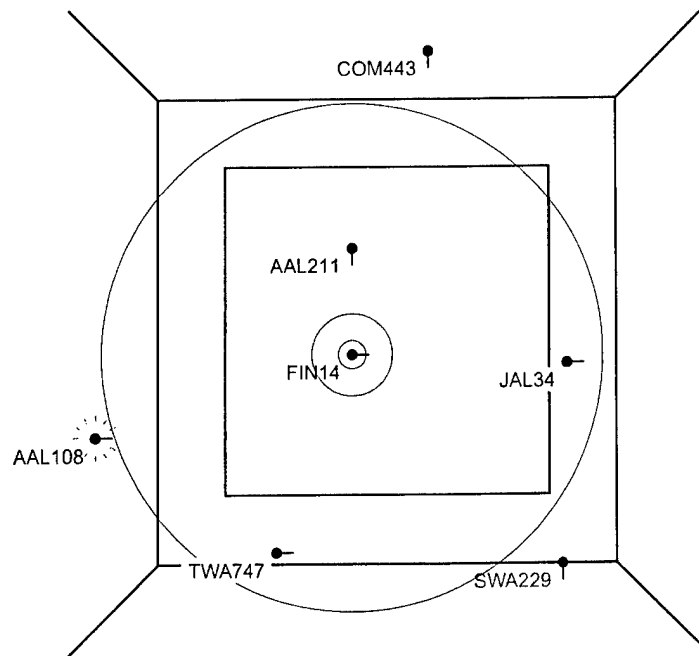
Figure 11: Two examples of the computation of CLOSEST-BLIP-IN-FRONT-IN-LINE.

Our approach to computing CLOSEST-BLIP-IN-FRONT-IN-LINE can lead to the performance errors. Recall that the length of the bounding box is the outer radius of the parafovea (θ_{PF}) and that this radius is sampled from a uniform distribution on each saccade. If θ_{PF} happens to be on the low end of its distribution (resulting in a smaller parafovea) when the model is extracting the CLOSEST-BLIP-IN-FRONT-IN-LINE feature, then the bounding box may be too short to include an aircraft that is actually in front of and inline with the requesting aircraft.

For example, if the outer radius of the parafovea in the first example in Figure 11 were reduced by at least 20%, JAL34 would no longer be in the bounding box and CLOSEST-BLIP-IN-FRONT-IN-LINE would be FALSE. The model would immediately accept the speed request. In this case, accepting the speed request would be an error.

A short bounding box does not always lead to a flawed decision making in the DECIDE-SPEED-REQUEST operator. Suppose that the parafovea size in the second example were reduced by at least 25%. In this case, AAL211 would not be considered in front and inline with FIN14 and its CLOSEST-BLIP-IN-FRONT-IN-LINE feature would be FALSE. As before, the model would accept the speed request. However, in this situation, accepting the speed request is *not* an error since the aircraft are traveling in different directions. Therefore, errors from flawed decision making are very rare, occurring only in 1 of 4 situations where the length of the bounding box happens to be on the low end of its distribution. We have only observed this kind of error once or twice.

Two comments about the performance of task action sequences should be made. First, upon completion of the model, we noticed a difference between how our model was programmed to perform task action sequences and how we suspect subjects actually perform these sequences. The left sequence of Figure 12, the one used in the model, starts by looking, pointing, and clicking on the appropriate command button—L(COMMAND), P(COMMAND), C(COM-MAND)—followed by looking, pointing, and clicking on the aircraft, and so on for the ATC and the SEND button. All of these steps are performed sequentially. However, if we apply the ideas performance improvement through strategy refinement (Chong, 1998), we would expect experienced subjects to actually perform some of these steps concur-rently. As shown on the right of Figure 12, the manual and ocular motor actions could be performed in parallel—the clicking (a manual motor behavior) and looking (an ocular motor behavior) steps are overlapped. EPIC accommo-dates this kind of concurrency since its motor processors are independent and can therefore execute behaviors in par-allel. The improved sequence requires less time to complete—the amount of time needed to do three mouse clicks; on the order of one second.

A second comment on how our task action sequences are performed pertains to an assumption we made about task action performance. Early in training, subjects can be observed doing what we will call "confirmatory glances." For example, after an object (an aircraft or ATC) has been clicked, the subject may look up at the command composition region (the area above the radar display) to confirm that the name of clicked object does indeed appear in the com-mand. As seen in Figure 12, we have not included these saccades or the necessary cognitive steps to do the confirma-tion. We have assumed that experienced subjects do not perform these extra steps. We can justify this assumption because when an object is pointed to, it is highlighted by the task environment. This highlighting is sufficient to con-firm to the user that the desired object is under the mouse.

7.6   OPERATOR: SEARCH-RADAR-DISPLAY

Assumptions about human behavior

- Memory creation/reconstruction of an aircraft's location
- Search biased to relevant regions of the display
- Search scan not always systematic
- Blips can be fixated more than once

Due to perceptual and memory limitations, it is not uncommon for the model to either forget the location of an aircraft, or to not know the location of an aircraft because it was never fixated. This condition only becomes a prob-lem when a task action needs to be performed.

This last operator, SEARCH-RADAR-DISPLAY, is used to find an aircraft given its IDENT. This operator is only acti-vated when a task action operator has impassed (or stalled) because the location of the aircraft was unknown, prevent-ing the composition of the appropriate message.

For every blip that is fixated as the search is performed, the model creates a *reasoned* memory called a MOST-RECENTLY-SEARCHED-BLIP memory; see Figure 5. Only blips that *do not* have a MOST-RECENTLY-SEARCHED-BLIP

25

CURRENT SEQUENCE   FASTER SEQUENCE

L(COMMAND)   L(COMMAND)

P(COMMAND)   P(COMMAND)

C(COMMAND)   C(COMMAND)   L(AIRCRAFT)

L(AIRCRAFT)   P(AIRCRAFT)

P(AIRCRAFT)   C(AIRCRAFT)   L(ATC)

C(AIRCRAFT)   P(ATC)

L(ATC)   C(ATC)   L(SEND)

P(ATC)   P(SEND)

C(ATC)   C(SEND)

L(SEND)

P(SEND)

C(SEND)

Figure 12: Comparison of the time to perform a task action sequence. The current sequence (left) used in the model does not exploit the ability to concurrently perform actions for different motor modalities as is seen in a faster sequence (right). L(), P(), C() refer to the overt actions of looking, pointing, and clicking, respectively.

memory are eligible for fixation by the SEARCH-RADAR-DISPLAY operator. If the aircraft is not the target, then the model recreates or rehearses the memory of the non-target's location and chooses another blip to fixate.

The model uses the list of most recently searched blips to try to prevent blips from being refixated, allowing only unfixated blips to be examined. However, because the MOST-RECENTLY-SEARCHED-BLIP memories reside in the volatile memory partition, refixations can still occur. Refixation will generally occur when there are many blips on the display and several fixations have transpired but the target has yet to be found. With this list representation, the model makes the intuitive prediction that refixations are more likely to occur as the length of the search increases and the length of the search increases as the number of candidate aircraft increases.

Additionally, the SEARCH-RADAR-DISPLAY operator narrows the number of possible candidate aircraft by taking into consideration the impassed task action operator and the location of blips on the display:

- If the model is trying to perform a CONTACT-ATC task action, then the model knows that the aircraft must be in the transfer region.
- If the model is trying to perform an ACCEPTING-AC task action (accepting an inbound aircraft), then the model knows to only search for aircraft outside the center airspace; IN-CENTER-SPACE = FALSE.
- If the model is trying to perform a WELCOME-AC task action, then the model knows that the target blip will likely be in the center airspace; IN-CENTER-SPACE = T.
- If the model is trying to do a DECIDE-SPEED-REQUEST, then the target blip will likely be in the center airspace; IN-CENTER-SPACE = T.

26

The combination of forgetting and of narrowing the candidates based on blip features produces a search pattern that satisfies many of our assumptions about the behavior of searching the radar.

7.7 MULTI-TASKING

When running, our model first selects one of the three operator classes—MONITOR-RADAR, SEARCH-MHL, and TASK-ACTIONS—and then executes an appropriate operator within the class. There are often times when the preconditions of more than one operator class is satisfied. In these cases, there needs to be some knowledge for determining which class should be selected. A key component of the model is the arbitration of these three competing tasks classes.

Our model accomplishes this by what we will call a *reactive executive process*. We use the term *reactive* because our executive process (a mechanism that arbitrates among competing processes for access to limited resources) is a collection of reactive rules that are constantly monitoring the situation. There is no deliberate reasoning mechanism, such as an operator, that determines which class should be selected.

7.7.1 BETWEEN-TASK COMPETITION

At the level of competition *between* task classes, our executive operationalizes the priorities that emerge from a functional consideration of the competing tasks. In general, the task class priority is as follows: MONITOR-RADAR < SEARCH-MHL < TASK-ACTIONS.

The rationale for this ordering is based on the fact that the fundamental goal of this task is to minimize the performance score. The score is minimized by transacting aircraft. Therefore, our priority ordering, from lowest to highest, reflects the increasing imminence of performing a task action. MONITOR-RADAR is the lowest priority because it performs no actions and its purpose is to find something to do. This task class has two outcomes: finding a blip that (a) *might* need to be transacted or (b) *definitely* needs to be transacted. In the latter condition, a TASK-ACTION can be immediately performed, therefore this has the highest priority. In the former condition where a task action might need to be performed, the SEARCH-MHL class is performed. Since this task class may or may not lead to a transaction, it is given a priority between the other task classes.

7.7.2 WITHIN-TASK COMPETITION

Another form of competition occurs *within* task classes. Once a task class is selected, then, depending on the class, other arbitration rules will determine which operator within the class is selected for execution, assuming that more than one operator is vying for attention.

In the MONITOR-RADAR class, the MONITOR-RADAR-FIXATE and MONITOR-RADAR-THINKING operators are sequential and never compete with one another.

In the SEARCH-MHL class, the proximity-driven and onset-driven operators can compete with one another. The model assumes that these operators are equally important and will arbitrarily select one when there is competition. Additionally, there can be competing instances of the onset-driven operator. This occurs when two or more message onsets occur, either on a single message history list or distributed across the message history lists. For example, if a new message appears on both the inbound and outbound message history lists, then there will be two instantiations of the SEARCH-MHL-ONSET-DRIVEN operator. In this case, the model's priority scheme is as follows: speed-request < inbound < outbound.

The rationale for this ordering is derived from the scoring penalties stated in the task instructions. The penalty for delaying a response to a speed request message is very low. Additionally, there is no threat of an aircraft entering a hold pattern if the request is not handled. Therefore, speed-request messages have the lowest priority.

Inbound messages have a higher priority than speed request messages because there is an inbound message type— ACCEPT-REQUEST, e.g., ACCEPT KLM913—that will lead to an inbound aircraft entering a hold pattern if the message is not handled in a timely fashion.

The reason outbound messages are assigned the highest priority is as follows. Recall that it takes two transactions to fully process an outgoing aircraft. The first transaction (TRANSFERRING-AC) occurs when an aircraft crosses the inner border of the airspace display. A response from the destination ATC (e.g., SOUTH ACCEPTS AW610) to the TRANSFERRING-AC transaction will take some amount of time. During this delay, the outbound aircraft continues to

27

Figure 13: Block diagram of the model for the aided condition.
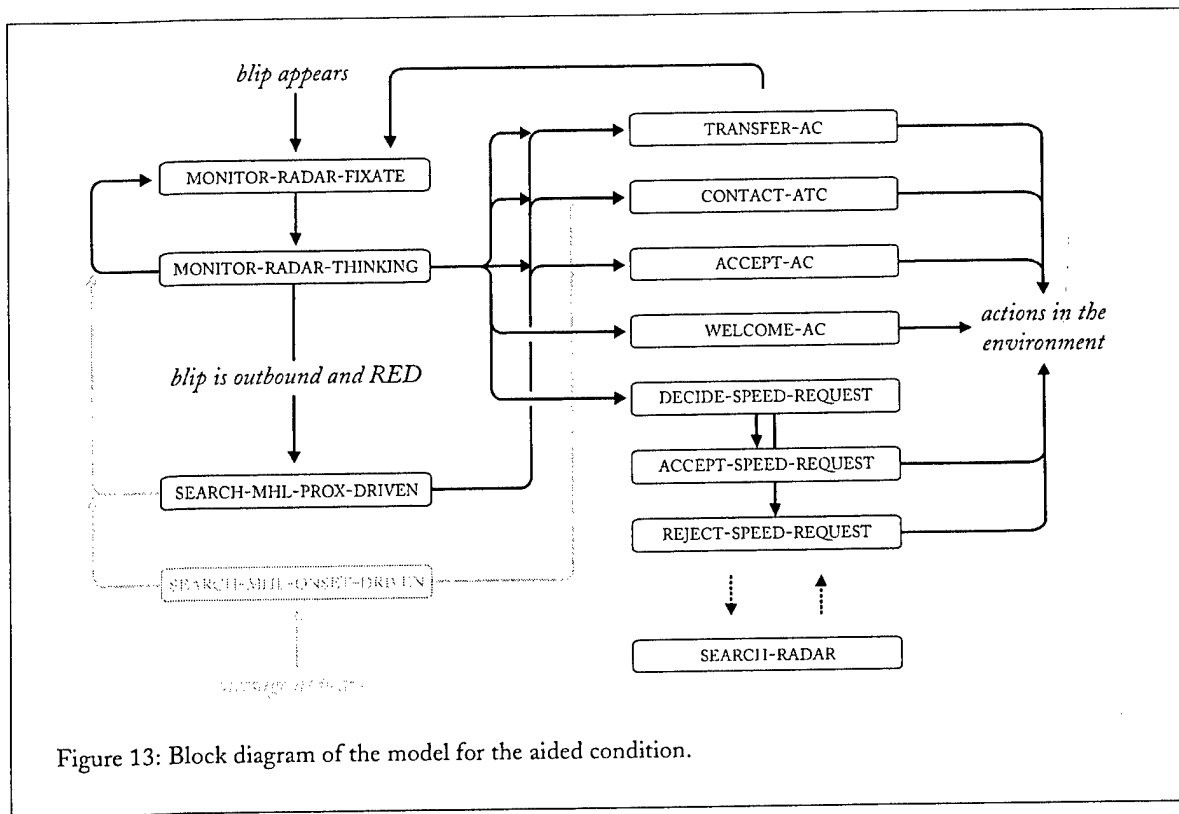
move towards the border of the airspace. By the time the destination ATC responds, the aircraft can be quite close to the border. To avoid entering a hold pattern, the second transaction (CONTACT-ATC) must be performed immediately in response to the acknowledge message.

The urgency of performing CONTACT-ATC is particularly strong in the typical case where there is a delay in performing the TRANSFERRING-AC task action; i.e, the model was not observing the aircraft at the moment it crossed the border, but noticed that the aircraft had already cross the and is already in the transfer region. This delay makes it more likely that the aircraft will be especially close to the border when the destination ATC acknowledges the transfer request.

Therefore, the reason we have given outbound messages priority over inbound messages is because outbound aircraft will be at greater risk of entering a hold pattern when the acknowledgement message is received than an incoming blip will be at the time it makes its request for acceptance into the center airspace.

Finally, there will always only be one task action to perform in the TASK-ACTION class.

8.0 THE MODEL IN DETAIL: AIDED (COLOR) CONDITION

The preceding description of the operators in the model constitutes an implicit description of the model used for the unaided condition. The model for the aided condition is illustrated in Figure 13. As can be seen, the model is primarily a subset of the unaided model found in Figure 6. A modified model was needed because although the unaided condition model could perform the task in the aided condition, it could not take advantage of the aspects of the aided condition that make it "aided" and would not have produced data representative of subjects in the aided condition.

We now briefly address each operator, highlighting how each was changed to reflect behavior in the aided condition.

28

- If (blip color is CYAN) then PBP = 8
- If (blip color is MAGENTA) then PBP = 12
- If (blip color is GREEN) then PBP = 14
- If (blip color is ORANGE) then PBP = 16
- If (blip color is YELLOW) then PBP = 18

Figure 14: Rule for assigning perception-based-priority to blips in the unaided condition.

8.1 OPERATOR: MONITOR-RADAR-FIXATE

We hypothesized that eye scan patterns in the aided condition would differ from that in the unaided condition. In the aided condition, a subject need only scan the display for colored (non-white) blips. We instantiated this hypothesis by assuming that perceptual properties *alone* are sufficient to produce reasonable scan patterns; i.e., knowledge-based priorities were not needed.

As in the unaided condition, blips were assigned a perception-based priority (see Figure 7) based on their perceptual proximity to the border of the airspace. Those PBP assignments applied only to WHITE blips (with the exception a RED blip). For the aided condition, we extended those PBP assignments with five (5) new color-specific assignments, listed in Figure 14.

The determination of the most important blip to fixate was based solely on PBP. The selection of the blip to fixate was computed as before, but only rule C (from Section 7.1) was used.

## 8.2 OPERATOR: MONITOR-RADAR-THINKING

Figure 15 illustrates the levels of processing used in the aided condition. Since MONITOR-RADAR-FIXATE did not need knowledge-based priorities (KBP), almost all the memories on which it depended—EXPECTATION, ANTICIPATION, PROXIMITY, IN-OR-OUT, and COMMANDS—were no longer needed. The end result is that thinking in the aided condition needed to be only two (2) levels deep, in contrast to five (5) levels in the unaided condition. The result is that the model spends less time thinking about each blip.

8.3 OPERATOR: SEARCH-MHL-PROXIMITY-DRIVEN

Invocations of proximity-driven searches are significantly reduced in the aided condition because as long as an aircraft is WHITE, the model knows that all that could be done to the aircraft has been done. The only exception, as indicated in Figure 13, occurs if an outbound blip turns RED. Recall that outbound blips require two transactions in order to exit the center airspace. If an outgoing aircraft is RED, the model cannot determine which transaction/s have been missed. Therefore, the outbound MHL must be searched to determine what action/s are missing in the case that an outbound blip turns RED.

8.4 OPERATOR: SEARCH-MHL-ONSET-DRIVEN

Figure 13 illustrates that onset-driven searches are completely eliminated in the aided condition. The primary purpose of the reading the MHL when onset occurred was to find messages that prompted a task actions. However, in the aided condition, this functionality is indicated by aircraft color changes.

Figure 15: Functional dependency used by MONITOR-RADAR-THINKING in the aided condition
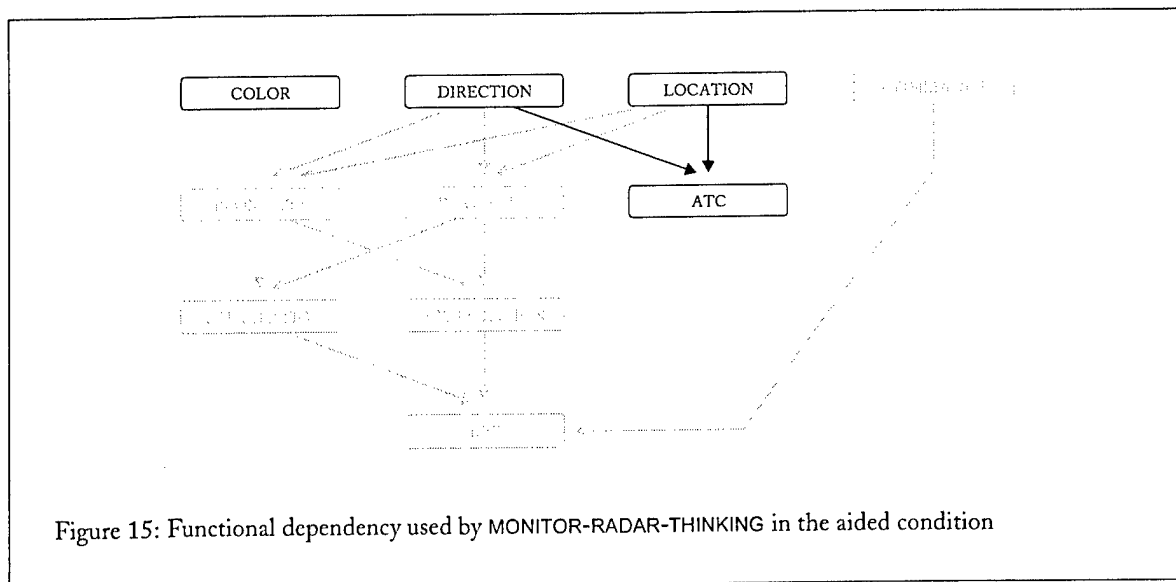
8.5 OPERATOR: TASK-ACTIONS

All task action operators remain the same.

8.6 OPERATOR: SEARCH-RADAR-DISPLAY

The SEARCH-RADAR-DISPLAY operator remains in the model.

8.7 MULTI-TASKING

Multi-tasking is the same as in the unaided condition.

9.0 MODEL SUMMARY

A framework for understanding the changes that appear in the aided model is to realize that a task condition is called "aided" because the task consumes fewer human resources. Examples of resources that might be conserved are: ocular motor time, working memory, cognition (reasoning), and manual motor time. For this task, the aided condition reduces the requirements of the first three resources.

Ocular motor time (scanning effort) is dramatically reduced because the model does not have to read message *onsets* to determine what actions to perform. Also, the model does not have to perform proximity-driven searches for non-RED blips. The model still has to do proximity-driven searches for outbound blips that are RED, but these instances are greatly reduced since the overall task is easier.

Working memory consumption is also dramatically reduced because the model does not need to create or maintain many of the memories that were required in the unaided condition. For instance, in the unaided condition, the model needed to create and maintain knowledge-based priority (KBP) memories. Since KBP is no longer needed, most of the memories on which KBP depended are also unnecessary: EXPECTATION, ANTICIPATION, PROXIMITY, IN-OR-OUT, and COMMAND memories were all unnecessary. This was illustrated in Figure 15.

30

Workload (t) = $\alpha * \Sigma v / t$

TASK TO BE DONE	AIDED CONDITION	UNAIDED CONDITION	VALUE
unknown	Red	Red	10
transferring-ac	Orange	Anticipation	3
contact-atc	Yellow	Expect acceptance message	4
accepting-ac	Green	Expect accept request	3
welcome-ac	Cyan	Expect welcome request	1
accept/reject request	Magenta	Speed request	2

Figure 16: Tasks and associated workload costs.

Finally, cognitive resources are also conserved in the aided condition. Figure 15 illustrates that thinking about an aircraft in the aided condition only requires two (2) processing levels, in contrast to five (5) in the unaided condition.

10.0 WORKLOAD COMPUTATION

Part of the human data collected for task was subjective workload evaluations. One challenge of the modeling effort was to devise a means by which the *model* could produce *subjective* workload measures. There is a very large space of possible parameters that could impact such a workload rating.

Our implementation derives workload from the *realization* that there is a task to be done. When the model notices there is work to be done, it adds a "cost" factor to a cumulative sum of costs (Σv). At the end of the model run, the total score is divided by the duration of the scenario (t) and multiplied by some scaling factor (α). The equation is shown in Figure 16. Our formulation of workload captures two intuitive aspects of workload: (a) workload is a function of the work done per unit time, and (b) workload is also a function of the difficulty or urgency of the work done per unit time.

Although the triggers for pending tasks differ in the aided and unaided task condition, the procedure for computing workload remains the same. Figure 16 shows the lists of task actions in the first column. The fourth column lists the "workload cost" associated with each task. (These costs are a fabrication that correlates to the urgency of or importance of the task.) The middle columns list the events, by task condition, that cause the realization that there is a task to be done.

In the aided condition, this realization occurs when an aircraft changes from WHITE to some other color. In the unaided condition, an aircraft's color only triggers a cost when an aircraft is RED. Otherwise, the realizations occur when anticipations and expectations are *created*. (Speed requests cannot be anticipated or expected, so their realizations occur when a speed request appears.) Recall that anticipations and expectations are subject to forgetting. If they are forgotten and recreated, then the associated cost is again added to the cumulative workload sum.

The primary distinction between the computation of workload in the aided and unaided condition is that in the aided condition, each pending task causes a *single* increment of the cumulative cost. In the unaided condition however, each task per blip can produce *multiple* increments of the cumulative cost through the creation and re-creation of forgotten memories.

Our approach to computing workload captures the main effect of the task condition on subjective workload: workload scores for the unaided condition are higher than in the aided condition.

Generally, performance scores are positively correlated with workload scores. Examples of this correlation could be found in the subject data. Our subjective workload formulation was able to produce positive correlations. To our surprise, however, our formulation also produced *negative* correlations: the model's subjective report was low while the performance was high, and vice versa.

We were unsure if this was a bug or a feature of our formulation. We are cautiously inclined to believe it is the latter since it is known that self-evaluation of performance does not always correlate positively with actual performance.

Gopher & Donchin (1986) state: "The overall correspondence between subjective measures [of workload] and performance measures is not very high, with equal proportions of positive and negative reports from experimental work....It is possible that the subjective estimate captures an objective and external assessment of the task. It does not, however, serve to predict how well, or how poorly, a subject will do. One may infer from this observation that the responses that subjects give when asked for an estimate of the difficulty of tasks are based largely on a "cognitive" analysis of their knowledge of the task rather than on an assessment of the way they actually interacted with the task."

11.0 MODEL PERFORMANCE

Our model has many stochastic components found at the perceptual, memory, cognitive, and motor levels:

- At the perceptual level, the sizes of the fovea and parafoveal retinal zones are varied between saccades. This is done because there are no hard retinal zone boundaries in the human eye, so we vary the radii to "fuzzy up" the boundary between these zones. Additionally other perceptual parameters (feature transduction times) are stochastically varied between trials as a normal characteristic of EPIC.

- There are two important memory parameters associated with memory; the parameters that affect the rate at which working memory elements are forgotten: BLA-THRESHOLD and BLA-DECAY-RATE. These values were *not* modified within or between trials in our model. However, it might be the case that these should actually be varied, at least between trials.

- At the cognitive level, the duration of the cognitive cycle varies uniformly between 33 and 67 ms and changes between trials Also, there are times when the model arbitrarily chooses an operator among a set of competing candidates. For example, if SEARCH-MHL-PROXIMITY-DRIVEN and SEARCH-MHL-ONSET-DRIVEN are both executable, then the model arbitrarily selects between the two.

- At the motor level, various performance parameters are stochastically varied between runs.

Because of this non-determinism, we view each model run as representing one individual's performance on a single trial. To evaluate the model would ideally require many runs (at least as many runs as there were human subjects) per condition, generating a distribution from which mean and variance statistics could be computed.

Our model ran much slower than real-time and prevented us from running as many trials as we would have liked. In the end, we had to settle for one trial per condition. This was very unfortunate because very few conclusions could be made about the model's performance relative to the confidence intervals that were defined by the Moderator.

On the other hand, one crude approach to evaluate the model's performance is to determine whether or not the trial-by-trial performance in a task condition falls within the *distribution* of performance for subjects in the same task condition. The table in Figure 17 shows one set of data from model runs on the tuning data.[11] Though not complete (more runs are need in several conditions), it does show that individual model runs do tend to fall within the range of performance for subjects.

While the model runs do fall within the distribution of human performance, the model does not produce representative means or variability as seen in the subject data. For example, in the unaided condition, the lowest model score is much higher than the lowest human score. We suspect that this discrepancy might be remedied by stochastically varying (between trials) the parameters that affect the rate at which working memory elements are forgotten: BLA-THRESHOLD and BLA-DECAY-RATE. We would expect that a lower threshold (a more negative value) and/or a smaller decay rate would result in less forgetting which would lead to better memory performance, and possibly better overall performance on the task. Conversely, we expect that a higher threshold and/or a larger decay rate would result in more forgetting and worse overall performance.

Yet another evaluation of the model that could have been done was an evaluation of the *overt* performance of the model. This approach is best suited to architectures and modeling approaches such as ours that represent eye and hand movements. For instance, we have a very compelling graphical demonstration of our model that shows the eye scanning the radar display, reading the message history lists, and the coordination of eye and hands as task actions are

---

[11] "Tuning" data and scenarios were provided to the modelers for model development. The final model evaluation was based on a new set of data and scenarios.

TASK CONDITION	HUMAN DATA	MODEL DATA
Aided-Low	0, 0, 0, 0, 0, 0, 0, 10 ($\bar{x}$ = 1.25, σ = 3.53)	0, 0 ($\bar{x}$ = 0, σ = 0)
Aided-Med	0, 0, 0, 0, 0, 0, 10, 30 ($\bar{x}$ = 5, σ = 10.69)	0, 0 ($\bar{x}$ = 0, σ = 0)
Aided-High	0, 0, 0, 0, 0, 0, 10, 24 ($\bar{x}$ = 4.25, σ = 8.71)	0, 0 ($\bar{x}$ = 0, σ = 0)
Unaided-Low	0, 0, 0, 5, 10, 11, 60, 70 ($\bar{x}$ = 19.5, σ = 28.54)	46, 50, 50 ($\bar{x}$ = 48.67, σ = 2.31)
Unaided-Med	0, 4, 6, 57, 140, 152, 208, 289 ($\bar{x}$ = 106,87, σ = 107.83)	54, 63, 101, 101, 132, 149, 149, 200, 213 ($\bar{x}$ = 129.11, σ = 55.29)
Unaided-High	106, 131, 131, 224, 417, 421, 446, 674 ($\bar{x}$ = 318.75, σ = 202.57)	228, 294, 370, 381,382, 623, 649, 1050 ($\bar{x}$ = 497.125, σ = 267.57)

Figure 17: Comparison of human and model performance on the tuning scenarios.

performed. When viewing the performance in this manner, one can evaluate whether the performance *appears* valid; this is essentially a limited Turing test. Certainly, a more rigorous and objective analysis would be preferred, i.e., collecting eye tracking data, syncing this data to mouse movements, then determining statistically if the model's overt behavior was qualitatively similar to that of humans.

12.0 A MEMORY-BASED EXPLANATION OF STRATEGY PREFERENCE IN THE COMPOSITION OF MESSAGES
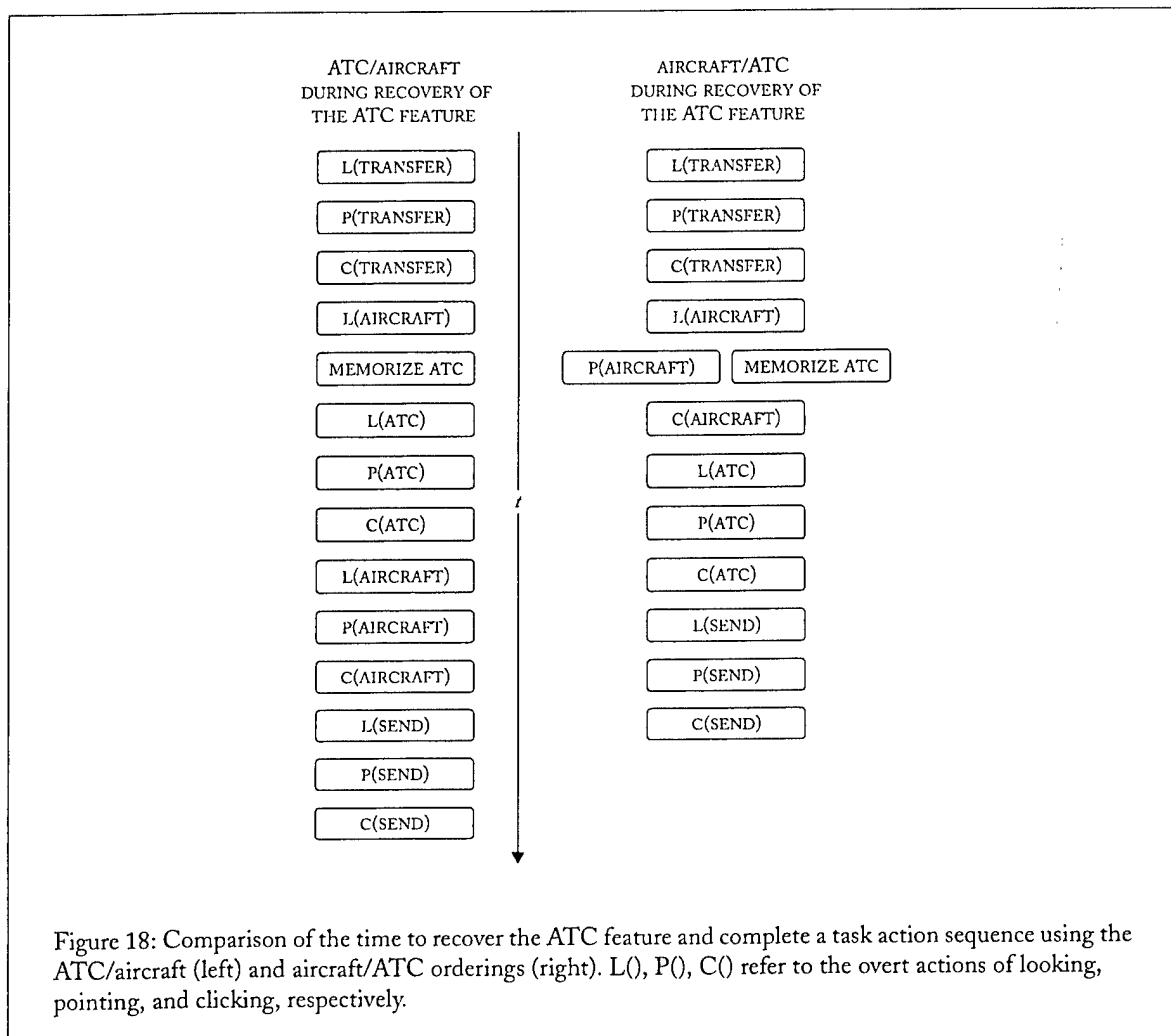
When composing a message, say sending a TRANSFERRING-AC command for JAL34, the subject (and the model) must specify both the ATC and the aircraft. In a preliminary version of the task environment, there was a requirement that the ATC be specified before the aircraft, i.e., click the TRANSFERRING-AC button, click on the ATC, click on the aircraft, then click on the SEND button. In the final version of the task environment, this requirement was removed, permitting either ordering—ATC/aircraft or aircraft/ATC.

At the modeler's workshop (where the removal of the ordering constraint was announced), there was a discussion of the methodological problems this removal might introduce. While beneficial to the subjects (presumably because it allowed subjects to use the strategy that best suited their performance style), it introduced a degree of freedom; subjects could randomly choose one strategy over another, or they could consistently prefer one strategy over another.

The random strategy would be easy to model and might not require further explanation of why subjects performed in this manner. However, if subjects were to consistently choose one strategy over another, one would want/expect a model to explain why the selected strategy was preferred.

There were anecdotal reports from both the subjects and most modelers who performed the task that the aircraft/ATC ordering was generally preferred, and in some, preferred exclusively. However, no one could explain why this might be the case.

An early iteration of our model exclusively used the ATC/aircraft ordering strategy as required in the preliminary task environment. Since it was not clear why subjects would prefer the alternate ordering, we kept the ATC/aircraft ordering in our model. Upon reaching a point in the model development where we felt the performance was adequate, we turned to the task of extending the Soar architecture to accommodate working memory decay, as discussed previously in Section 5.

ATC/AIRCRAFT DURING RECOVERY OF THE ATC FEATURE	AIRCRAFT/ATC DURING RECOVERY OF THE ATC FEATURE
L(TRANSFER)	L(TRANSFER)
P(TRANSFER)	P(TRANSFER)
C(TRANSFER)	C(TRANSFER)
L(AIRCRAFT)	L(AIRCRAFT)
MEMORIZE ATC	P(AIRCRAFT) MEMORIZE ATC
L(ATC)	C(AIRCRAFT)
P(ATC)	L(ATC)
C(ATC)	P(ATC)
L(AIRCRAFT)	C(ATC)
P(AIRCRAFT)	L(SEND)
C(AIRCRAFT)	P(SEND)
L(SEND)	C(SEND)
P(SEND)	
C(SEND)	

Figure 18: Comparison of the time to recover the ATC feature and complete a task action sequence using the ATC/aircraft (left) and aircraft/ATC orderings (right). L(), P(), C() refer to the overt actions of looking, pointing, and clicking, respectively.

When we subsequently applied working memory decay to our model, we immediately discovered that on occasions where the model was composing a message and had to specify the ATC, it was too frequently unable to recall the ATC memory associated with the aircraft. As a result, the model broke.

Initially, we considered this to be a good demonstration that the decay mechanism was working properly and could produce the kind of forgetting observed in subjects. Also we felt it highlighted areas in our model where we needed to introduce a recovery strategy for coping with volatile memory. The recovery approach we used (one that subjects probably also use) is as follows: if the ATC memory cannot be recalled, then look at the aircraft to extract the ATC feature, thereby recreating the memory. Once the memory is recreated, the model can then recall the ATC memory and will click the ATC, followed by the rest of the message—clicking on the aircraft, then clicking on the SEND button.

However, although this recovery approach worked beautifully, it missed the real issue. Adding the recovery strategy just meant that the model was now too frequently reconstructing the ATC memory. The real issue was that the model was unable to remember the ATC memory.

The reason why the ATC memory is so readily forgotten can, in part, be deduced from Figures 10 and 15. These figures show that no memories are dependant on the ATC memory. The result is that the ATC memory is not used (i.e., does not get its activation increased) to create other memories. Additionally, an inspection of our model (and a functional analysis of the task) revealed that the ATC memory is *only* necessary when a command is being composed.

We then realized why the aircraft/ATC composition strategy might be preferred (see Figure 18). When the model performs a recovery, it has to look at the aircraft, extract the ATC feature, look at the ATC, click on the ATC, look

back at the aircraft, click on the aircraft, then click on the SEND button. In contrast, the aircraft/ATC strategy does not require redundant saccades to the aircraft. The result is more streamlined and faster performance.

Our conclusions therefore are that subjects *may* prefer the aircraft/ATC ordering because it takes less time to perform. Although reaction-time is not the focus of the task, it is implicitly important since there can be a large cost for "falling behind" in this task. Subjects may be sensitive to this fact and hence prefer faster strategies as a result.

Also, subject may prefer the aircraft/ATC ordering because it involves less redundant steps, as seen in Figure 18 where the aircraft has to be looked at twice. It is conceivable that subjects are sensitive to these kinds of inefficiencies.

Another theoretical benefit of the aircraft/ATC ordering is that it reduces the demand on memory resources. When thinking about an aircraft, the subject need not create or be concerned about maintaining or reconstructing a memory of the ATC feature.

There may well be other, more straightforward, explanations of why one ordering would be preferred over another. Nevertheless, we are pleased that *any* explanation at all, much more, a plausible one, just "fell out" of the model given that we were not even pursuing an explanation for this strategy preference. This explanation is purely is a product of low-level modeling with realistic models of human perception and memory capabilities and their limitations.

13.0 CONTRIBUTIONS

This effort contributes several important changes to the architecture and opportunities for further research:

- Prior to this work, EPIC-Soar had only been applied to a simple single and dual-task condition (Chong & Laird, 1997; Chong, 1998). This effort provided a task that is significantly more dynamic and complex and demonstrated the efficacy of this hybrid architecture.
- This task demonstrated the importance of memory limitations. Prior to this effort, there has been a glaring disconnect between the behavior of Soar's declarative memory and long-held theories and observations of the behavior of human memory. This effort provided the motivation and funding to implement a form of base-level activation (as found in ATC-R) within Soar. We have been able to demonstrate that a volatile, more human-like declarative memory can be used to produce observed variances in human behavior and errors in behavior, and can be leveraged to provide plausible explanations of strategy selection.
- The addition of base-level activation draws Soar and ACT-R closer together, finally providing an opportunity for meaningful comparison studies to be made between the architectures.

14.0 REFERENCES

Anderson, J. R. & Lebiere, C. (1998). *Atomic components of thought*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Chong, R. S. (1998). Modeling dual-task performance improvements: Casting executive process knowledge acquisition as strategy refinement. Doctoral dissertation, Department of Electrical Engineering and Computer Science, The University of Michigan.

Chong, R. S. & Laird, J. E. (1997). Identifying dual-task executive process knowledge using EPIC-Soar. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Gopher, D. & Donchin, E. (1986). Workload—An examination of the concept. In K. R. Boff, L. Kaufman, & J. P. Thomas, (Eds.). *Handbook of Perception & Performance, Volume II: Cognitive processes and performance*. New York, NY: John Wiley & Sons.

Kieras, D. E. & Meyer, D. E. (1994). *The EPIC architecture for modeling human information-processing and performance: A brief introduction*. (EPIC Technical Report No. 1, TR-94/ONR-EPIC-1). Ann Arbor, University of Michigan, Department of Electrical Engineering and Computer Science.

Lewis, R.L., Huffman, S.B., John, B.E., Laird, J.E, Lehman, J. F., Newell, A., Rosenbloom, P. S., Simon, T. & Tessler, S. G. (1990). Soar as a unified theory of cognition: Spring 1990. In *Proceedings of the 12th Annual Conference of the Cognitive Science Society*.

Meyer, D. E. & Kieras, D. E. (1997a). *A Computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms*. Psychology Review 104 (1), pp 3-65.

Meyer, D. E. & Kieras, D. E. (1997b). *A Computational theory of executive cognitive processes and multiple-task performance: Part 2. Accounts of psychological refractory-period phenomena*. Psychology Review 104 (4), pp 749-791.

Miller, C.S. & Laird, J.E.(1996). Accounting for graded performance within a discrete search framework. *Cognitive Science*, 20(4), 499-537.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Pew, R. W. & Mavor, A. S., (Eds). (1998). *Modeling human and organizational behavior: Application to military simulations*. Washington, D.C.: National Academy Press.

Rosenbloom, P.S., Laird, J.E., & Newell, A. (1993). *The Soar Papers*. Cambridge, MA: The MIT Press.